FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Planeamento e Execução de Manobras de Veículos Aquáticos com Restrições de Movimento

Graça Cristina Carvalho e Silva

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Aníbal Castilho Coimbra de Matos

2 de Setembro de 2013

Resumo

Ao longo dos últimos anos, tem-se observado um crescimento exponencial da presença de sistemas robóticos na sociedade. Primeiramente motivada pela necessidade, surgiu a evolução robótica na área industrial com a utilização dos manipuladores industriais. Hoje em dia os investigadores têm vindo a concentrar esforços no desenvolvimento de robots móveis, os quais abrem portas a toda uma nova panóplia de questões e, posteriormente, de aplicações.

Um robot móvel é, de modo simples, de um sistema que deve ser capaz de se movimentar e interagir num ambiente definido.

Este trabalho apresenta um sistema que é responsável por automatizar a tarefa de condução de um veículo subaquático autónomo. O sistema permite controlar o AUV através da leitura de instruções de manobras que são definidas com recurso ao raio mínimo de curvatura que este consegue executar de modo a permitir ao veículo poder fazer uma acostagem ou inspeção do fundo aquático.

A execução da tarefa obrigou a estudar também algoritmos para realizar o planeamento de trajetórias para robots móveis: (i) uma topologia que faz a decomposição do espaço de configuração em células; (ii) campos potenciais artificiais; e (iii) pesquisas em grafos.

Contribuindo para o planeamento de caminho de um veículo autónomo, foi definido o algoritmo 1 e 2 que recorre a modelos matemáticos cinemáticos e dinâmicos que definem um veículo sujeito a restrições de não holonomia. O resultado destes algoritmos mostra que o AUV será acostado corretamente mesmo em ambientes cujo espaço disponível foi restringido.

O MARES é um destes AUV, desenvolvido no laboratório do *OceanSys*, pertencente ao INESC TEC na Faculdade de Engenharia da Universidade do Porto. A sua característica bastante particular de um sistema de propulsão desacoplado permite-lhe fazer movimentações em ambientes restritos com risco de colisão reduzido ao mínimo sendo, por isso, um veículo adequado ao projeto.

Abstract

Throughout the last few years there has been an exponential growth of the presence of robotic systems in society. Primarily motivated by necessity, the evolution of robotics in the industrial area has come about with the use of industrial manipulators. Nowadays, investigators have been focusing their efforts in the development of mobile robots, which pave the way to a whole new array of questions, and thus, possibilities.

A mobile robot is, simply put, a system that must be capable of moving and interacting in a known environment.

This paper presents a system that is responsible for automating the task of driving an autonomous underwater vehicle. The system allows to control the AUV through the reading of already defined and instructed maneuvers, by means of the minimum curvature radius that the vehicle can execute, in a way that allows the vehicle to dock or inspect the underwater bed.

To reach this point it was also necessary to study some algorithms to be able to create the planning of mobile robot trajectories: (i) a topology that decomposes the configuration space in cell divisions; (ii) artificial potential fields; and (iii) graph search.

Contributing to the path planning of autonomous vehicles, the algorithms 1 and 2 were defined by using cinematic and dynamic mathematical models which define a vehicle subjected to nonholonomy restrictions. The result of these algorithms show that the AUV will be docked correctly even in environments where the available space has been restricted.

MARES is one of such AUV, developed in the *OceanSys* laboratory, belonging to the INESC TEC in Faculdade de Engenharia da Universidade do Porto. It's quite peculiar feature of a propulsion system uncoupled allows it to make movements in restricted environments with the collision risk reduced to a minimum making it the adequate vehicle for the project.

Agradecimentos

Como qualquer projeto, esta dissertação, embora realizada a nível pessoal, não teria sido feita sem o apoio das pessoas que me rodearam no decorrer de todo o percurso. Como tal gostaria de deixar por escrito os meus mais sinceros agradecimentos:

Em primeiro lugar, gostaria de expressar a minha profunda gratidão ao meu orientador, o Professor Aníbal Matos, pelo tempo cedido, pela experiência, motivação e sugestões construtivas que partilhou comigo ao longo deste projeto. Sem ele, este, não teria sido possível.

Queria também deixar um agradecimento especial aos professores Armando Araújo, Maria Rosário e Paulo Costa que incutiram em mim o bichinho do curso, fazendo-me crer que o mesmo ia de encontro às minhas características pessoais, apesar das dificuldades que necessariamente apareceriam e da persistência que seria necessária para a vencer.

A todos os meus amigos que conheci na faculdade e a quem espero ter deixado um bocadinho de mim, tal como eles deixaram algo próprio em mim. Fica o meu desejo a todos de um futuro muito sorridente. De todos aqueles com quem trabalhei, gostaria de deixar um agradecimento especial àqueles que citarei, de seguida, pela companhia, partilha de opiniões, sugestões e momentos vividos durante esta aventura universitária que me conduzirá ao final desta etapa.

Ao Ricardo e à Margarida cujo companheirismo nas longas noites foi imprescindível, de onde surgiram muitas ideias e que me ajudaram sempre a levar as tarefas a bom porto.

À Diana, por ser uma inspiração para mim em palavras, ideias e ações.

Às demais Electras Elisabete e Mafalda pelas gargalhadas, as novas (e velhas) experiências e pela vossa amizade e carinho.

E, não poderia deixar de reconhecer, aqueles que se tornaram os meus primeiros companheiros de longa data: a Fátima, a Anabela, o Daniel, pois, enquanto houver pessoas como eles na minha vida, tudo é possível.

Não poderia deixar de agradecer à minha querida avó Aurora, pela sua preocupação e amor constante e, muito mais à Lígia, cujos conselhos, palavras amigas e ajuda em geral que me foram indispensáveis e muito preciosos.

E claro, agradecer todo o suporte que os meus irmãos Pedro e Luís e o apoio inquestionável dado pelos meus Pais que sempre acreditaram em mim no decorrer de todo o curso. Graças a eles tive acesso a muitas oportunidades únicas e fizeram-me crer que conseguiria chegar mais longe.

Graça Silva

"You are always a student, never a master.
You have to keep moving forward."

Conrad Hall

Conteúdo

Lista de Figuras

Abreviaturas e Símbolos

AUV	Autonomous Underwater Vehicle
ROS	Robot Operating Systems
DROS	Dave's Robotic Operating System
ASV	Autonomous Surface Vehicle
UAV	Unmaned Autonomous Vehicle
C-Space	Espaço de Configuração
MARES	Modular Autonomous Robot for Environment Sampling
UWSim	UnderWater Simulation
OSG	OpenSceneGraph
GPS	Global Positioning System
FIFO	First In First Out
LIFO	Last In First Out
OceanSys	Ocean Systems Group

Capítulo 1

Introdução

O robot é, cada vez mais, um companheiro de trabalho do homem que o ajuda a resolver alguns problemas práticos e tarefas complicadas, podendo, em determinadas situações, substituir o trabalhador.

Hoje em dia desenvolvem-se robots com variados formatos, sendo estes aparelhos objetos de muita investigação a nível industrial e universitário, tal como acontece neste caso. No meio do fantástico, várias são as referências na literatura ou nos *media* que apresentam os robots do futuro. Na realidade, já existe toda uma panóplia destes aparelhos que, como diz *Daniel H. Wilson* (famoso roboticista americano, que escreve colunas sobre a robótica), *"(...) são demasiado fantásticos para as pessoas acharem que é realidade."* De facto, existem robots que pedem empréstimo ao comportamento dos animais para caminharem sobre um coração vivo (como o robot *Heartlander*, cuja pesquisa se encontra disponível em [2]), quiçá o *octobot* que pode salvar a sua vida no oceano (o robot *Octopus*, apresentado em [3]) entre outros.

Em ambiente naval, os robots já são muito utilizados e estão muito desenvolvidos ao nível da marinha de guerra. Contudo, os estudos desenvolvidos ao nível da robótica, estão ainda muito aquém do desejável, relativamente à sua aplicação a nível de missões de socorro e aquisição de conhecimento de estruturas em meio aquático ou no fundo marinho.

Nesta dissertação irá ser abordado o planeamento e execução de manobras de veículos autónomos aquáticos com restrições de movimento, concebendo o projeto desde a conceção à sua implementação, que se espera viável em diferentes situações, sejam elas de ambiente industrial, laboratorial, missões de socorro, entre outras.

Com este projeto procura-se dar mais um pequeno passo no desenvolvimento e/ou melhoria de técnicas de planeamento de estacionamento em robótica aquática.

1.1 Motivação

Pode-se afirmar que, do ponto de vista comercial, o uso de veículos autónomos em ambiente marítimo é uma tecnologia ainda bastante recente, experimental. De facto, é uma área que só nos anos 90 começou a proliferar, pelo menos em grande escala, ao nível dos veículos autónomos

marítimos [4]. Apesar disso, a gama de possibilidades de aplicação destes veículos é extensa e traz vantagens para serviços considerados perigosos para o ser humano.

Como o ambiente marítimo não foi ainda explorado a fundo pelo homem, apresenta muitos desafios dentro e fora da área da robótica. Assim sendo, a autonomia e a automatização da robótica marítima é algo deveras almejado, pois permitirá uma atividade de navegação em tempo real com a vantagem de utilizar *hardware* acessível.

Adicionalmente, a tecnologia envolvida é complexa, mas sobretudo multidisciplinar, pelo que a navegação torna-se num grande desafio, tanto a nível de *software* como de *hardware*.

Assim, este trabalho em particular é uma aposta com perspetivas de futuro já que pode associar-se a missões de socorro, manutenção e recuperação de estruturas edificadas ou objetos, e torna-se um tópico muito presente e necessário na atualidade. Para além disso, a utilização destes aparelhos pode melhorar a navegação de veículos autónomos e tornar acessível e mais fácil as epopeias das explorações em meio aquático.

Com ele, pretende-se ajudar (de acordo com a investigação desenvolvida e experimentação laboratorial feita ao longo deste projeto) e contribuir, nem que seja com o avanço de um pequeno passo, para atingir os objetivos atrás referidos.

A nível pessoal, o trabalho desenvolvido, ao longo deste projeto, corresponde também à aplicação dos conhecimentos adquiridos ao longo do curso e, de alguma forma, a sua aplicação permitiu-me compreender melhor de que modo os posso pôr em prática num futuro próximo a nível profissional.

1.2 Caracterização do Problema

Uma das principais tarefas de um sistema de navegação de um veículo autónomo é a determinação da sua localização, ou seja, precisar a sua posição e orientação no ambiente de trabalho. O desenvolvimento da tecnologia necessária para robots autónomos engloba vários problemas importantes. Entre eles, encontra-se o planeamento de trajetórias.

À primeira vista, esta questão pode parecer simples, tal como refere *Mark W. Spong* [5], dada a nossa facilidade de movimentação no dia a dia. No entanto, tarefas simples como essa tornam-se difíceis de processar sempre que há restrições físicas, temporais ou geométricas, tais como: o desenho do robot, as condições do meio ambiente, o tempo disponível para atingir o objetivo pretendido ou, ainda, a forma dos obstáculos.

As principais dificuldades destes problemas estão associadas à existência de obstáculos e também a limitações de movimento do robot. De igual modo, as diferenças atrás referidas são transpostas para o meio aquático por via das restrições de espaço que o veículo terá que vencer, bem como pelas limitações de movimento deste em caminhos válidos.

Durante este trabalho revelou-se de particular interesse o estudo de um sistema iterativo de movimentação de um veículo aquático autónomo. A existência de problemas reais, nos quais a geometria do espaço de trabalho é apenas parcialmente conhecida no momento do planeamento da

trajetória, está largamente estudada (tal como foi referido por *D. Bodhale* para o caso do ambiente industrial [6]).

Assim, pretende-se que este trabalho implemente um sistema autónomo que seja capaz de fazer navegação num ambiente aquático que pode ser controlado, fazendo recurso dos seus sensores e a métodos de pesquisa de caminhos (por grafos, geometria computacional e outros).

1.3 Objetivos

Os esforços dirigidos para o trabalho da Dissertação partiram de alguns pressupostos identificados nos diferentes textos pesquisados, os quais serviram como ponto de partida do projeto e permitiram definir os objetivos aqui descritos:

- Definir uma manobra - visa o estudo dos diferentes movimentos a executar por um veículo aquático autónomo desde a sua partida até ao ponto de chegada;
- Identificar restrições à execução de uma manobra - partindo do princípio que o veículo se desloca numa linha reta quando não existem obstáculos físicos, temporais ou espaciais, aponta as restrições que são introduzidas no seu movimento, sempre que durante o percurso surgem obstáculos;
- Implementar manobras elementares - o estudo destas manobras visa conhecer a deslocação do veículo quando não há obstáculos no seu percurso.
- Implementar manobras múltiplas - estudo de manobras alternativas, mais complexas, para contornar obstáculos, com o menor custo computacional possível utilizando por exemplo, concatenações de manobras elementares;
- Simular a acostagem de um veículo autónomo num espaço restrito - recurso à simulação da deslocação do robot no mesmo percurso agora com obstáculos no ambiente do MatLab® .

1.4 Estrutura da Dissertação

Para além da Introdução, este relatório contém mais 5 capítulos. No capítulo 2, faz-se a apresentação do estado da arte e dos trabalhos relacionados, os quais resultam de pesquisa efetuada previamente à experimentação.

No capítulo 3 são referidos alguns modelos matemáticos, em particular o modelo cinemático do veículo autónomo. Para tal, foi tido em consideração as restrições geométricas, físicas ou temporais, as quais são tidas em conta na experimentação de forma a vencer as dificuldades encontradas.

No 4 é introduzida uma abordagem inicial da problemática do trabalho, assim como abordagens principais a desenvolver de acordo com as dificuldades encontradas, tal como foi referido anteriormente.

Segue-se o capítulo 5 onde são mostrados alguns resultados obtidos na fase de experimentação e uma breve análise do sistema implementado.

No capítulo final, 6, serão apresentados as conclusões obtidas a partir da experimentação efetuada e são referidos propostas de trabalho futuro a partir deste projeto.

Foi adicionado no anexo A da dissertação um conjunto de soluções possíveis para problemas de instalação de um simulador virtual numa máquina *OSX*, as quais foram obtidas através de experimentação própria e a interação com outros utilizadores internacionais assim como repositórios de versões. Atendendo a que todo este trabalho foi feito com recurso à língua inglesa, este capítulo apresenta-se neste idioma.

Capítulo 2

Revisão Bibliográfica

Neste capítulo reuniram-se algumas informações sobre o que foi desenvolvido até ao momento no âmbito do planeamento de caminhos de um robot móvel no seu ambiente de trabalho. Primeiro, é introduzida a robótica móvel na secção 2.1. Depois, faz-se uma breve referência ao módulo da localização em 2.2 e a sua aplicação em sistemas conhecidos da atualidade 2.3. Para se poder, então, estudar algumas soluções existentes da problemática do planeamento de caminhos na secção 2.4. Na secção 2.5 é abordado o problema da não holonomia que introduz dificuldade na prossecução dos objetivos a atingir durante a deslocação do veículo autónomo.

Por fim, são apresentados algumas vias opcionais de simulação em 2.6.

2.1 Robótica Móvel

Independente do meio (terrestre, aéreo ou submarino) em que operam, os robots móveis funcionam segundo princípios comuns e são suportados num conjunto de etapas básicas que vão ser mencionadas de seguida.

Outras funções que permitem habilitar os robots de autonomia na sua navegação são por exemplo os sistemas de visão computacional. Isto verifica-se tanto para o caso de se conhecer especificamente o meio em que o sistema se vai movimentar, como para o caso de o robot seguir num espaço que vai conhecendo de forma iterativa.

O autor *Qiang Zhan* [7] diz-nos que os três principais objetivos para um sistema automático de navegação completo são:

- a deteção de obstáculos existentes no percurso do robot;
- a medição da distância entre o robot e o(s) obstáculo(s);
- o planeamento de um trajeto viável.

Para que o robot possa cumprir a sua missão, executa, de um modo geral, um ciclo composto pelas seguintes fases:

$$\text{Perceção} \longrightarrow \text{Navegação} \longrightarrow \text{Atuação}$$

Qualquer que seja o sistema de locomoção, o robot move-se num ambiente que poderá ser dinâmico, quer este seja parcialmente ou totalmente conhecido. É durante a fase da **Perceção** que o robot obtém conhecimento do ambiente que o rodeia. De seguida, a fase de **Navegação** serve para o robot perceber onde está e para onde deve ir, planeando o caminho que deve descrever. Finalmente, na fase da **Atuação** é preciso fazer, de modo correto, o controlo devido dos atuadores para que o seu movimento seja eficaz tal como o planeado na fase de **Navegação**.

À medida que o robot se desloca, o ambiente circundante vai sendo diferente, pelo que, o ciclo de funcionamento do robot móvel está sempre em execução tal como mostra a Figura 2.1.

Figura 2.1: Modelo de um robot móvel [1]

2.2 Localização

Um dos aspetos mais importantes para a robótica móvel autónoma é o estudo dos métodos de localização, sendo, por isso, um dos alvos preferenciais dos estudos que se realizam na atualidade. Em [8], o autor *Johann Borenstein* expõe dois métodos para alcançar este objetivo: um que atua de forma absoluta e outro de forma relativa.

No caso da metodologia relativa, é conhecida a posição inicial do robot e estima-se a sua posição atual com recurso à odometria ou *dead-reckoning*. Esta metodologia, embora suscetível a falhas como a incrementação gradual de erros de computação ou outros fatores não sistemáticos, é de uso simples e barato. Por outro lado, a metodologia absoluta permite fazer a localização do robot sem o conhecimento, *à priori*, da posição inicial. Este modelo irá fazer uso de sistemas de localização externos a si tal como balizas ou qualquer outro tipo de marcos que auxiliem a navegar o robot em segurança.

Uma solução relativa pode muito bem servir de base para inicializar o método absoluto, mas, de modo geral, para se obter um bom resultado, uma boa prática será procurar combinar os dois métodos. Assim, sempre que são detetados os referidos erros de computação, é possível recorrer a um ponto de referência ou à utilização de outro método que permita atualizar a pose do robot de modo a este saber onde se encontra.

Em secções mais adiante 2.4, irão ser revistas algumas das técnicas comummente usadas para localização de robots móveis autónomos com recurso às duas metodologias.

2.3 Aplicações na Atualidade

Apresentam-se algumas das aplicações utilizadas na vida real onde se faz manobragem de veículos aquáticos autónomos. Estas podem ocorrer a nível militar, de investigação ou em missões de caráter geral. Seguem-se os exemplos:

- Acostagem (Figura 2.2), tal como o estacionamento do veículo em local próprio ou previsto para o efeito;

Figura 2.2: Acostagem entre embarcações [1].

- Navegação em zonas com constrangimento:
  - em limite de fronteiras nacionais ou zonas com a presença de radar inimigo;
  - na proximidade de cadeias de montanhas;
  - aquíferos ou redes de água fluviais;
  - junto a correntes marítimas, ou simplesmente, seguindo um cardume;
- Movimentações em locais de acesso complicado ao Homem (Figura 2.3);

[1] Imagem obtida em `http://waittfoundation.org/catalyst-program`
[2] Imagem obtida em `http://robotik.dfki-bremen.de/en/research/projects/eurex-1.html`

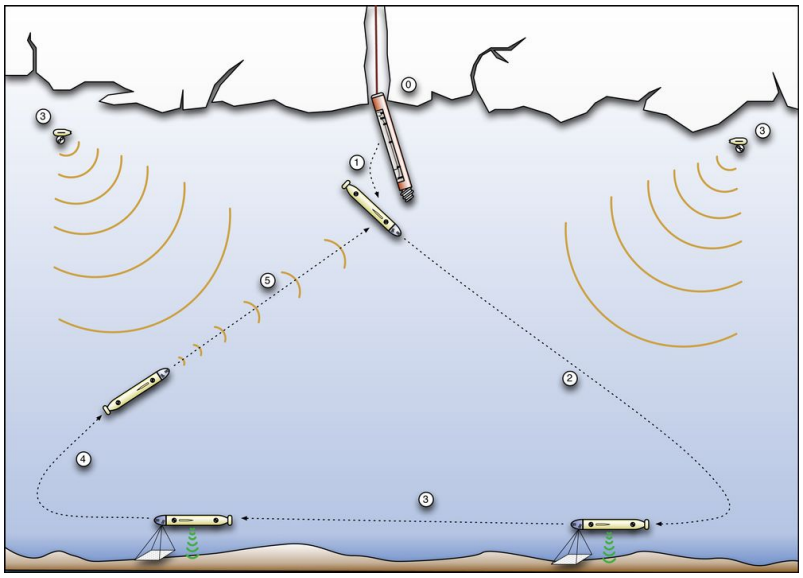


Figura 2.3: Manobras de um AUV em trabalho exploratório debaixo do gelo [2].

- Exploração geral do fundo subaquático (ver Figura 2.4) - estudo da geologia e/ou morfologia do fundo do mar e das suas costas;

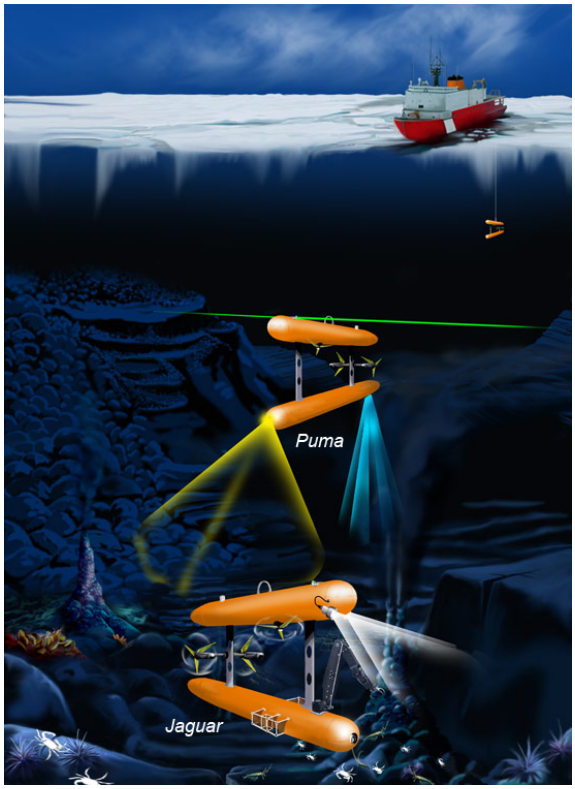


Figura 2.4: Exploração do fundo subaquático [3].

[3]Imagem obtida em `http://www.whoi.edu/oceanus/viewArticle.do?id=2459`

2.4 Planeamento de Caminhos

Encontrar um caminho que permita a um veículo autónomo movimentar-se de uma dada posição inicial até a uma posição objetivo, final, evitando eventuais obstáculos no seu ambiente de trabalho é o resultado que se pretende atingir.

O uso de termos como *caminho* e *trajetória* são muitas vezes confundidos um com o outro, quando utilizados de modo informal. No entanto, considera-se que a diferença substancial entre eles é que, o caminho é um conjunto de pontos que o robot pretende seguir evitando os tais obstáculos através da otimização de alguns critérios. Por outro lado, a trajetória, para além da vertente da geometria do caminho, toma em consideração também a dinâmica do veículo (velocidade, aceleração).

Na robótica, o problema de planeamento de caminhos pode ser bastante complexo, dependendo das restrições e parâmetros considerados para a sua resolução. A dificuldade de lidar com as restrições holonómicas (ver secção 2.5) inerentes aos robots móveis, a gama de parâmetros a ter em consideração como tamanho, forma e graus de liberdade, são exemplos de fatores que acrescem dificuldade do problema. Para simplificar a questão, é costume fazer-se uso do *espaço de configuração*, também conhecido como *C-Space*.

O problema torna-se então mais simples de resolver quando se pode modelar um corpo rígido complexo cujas dimensões são reduzidas às de um ponto. A configuração (de notação q) vai descrever a pose dos pontos do robot. O espaço de configuração C é simplesmente o conjunto de todas as configurações possíveis. A dimensão deste é dada pelo número k (de valor real) dos graus de liberdade: $q_1, ..., q_k$.

Considerando que o robot se move num espaço C com obstáculos conhecidos, a visualização do problema é mais simplificada (já que k pode elevar-se muito rapidamente). Determina-se um espaço de configuração dos obstáculos O como sendo um subespaço de C. Assim, pode-se simplesmente computar o espaço livre de colisões como sendo $F = C - O$ (tal como é descrito em [9]).

Para o propósito deste trabalho, o *C-Space* do robot que se move num plano com configurações a usar pelo robot do tipo $q = (x, y, \theta)$ é definido pela seguinte nomenclatura: $C = \mathbb{R}^2 \times SO(2)$ (onde *SO(2)*, é o grupo ortogonal de rotações 2D) [5].

Vários são os autores que apresentam algoritmos com a finalidade de resolver o problema descrito e, por exemplo [5, 10], dividem os métodos de abordagem do planeamento de caminhos em três grupos. Estes serão abordados nas subsecções 2.4.1- 2.4.3.

2.4.1 Campo Potencial

A metáfora que esta terminologia sugere é que o robot é uma partícula pontual no espaço de configuração, o qual se move sob a influência de um gradiente ou campo artificial de potencial *U(q)*. Este é construído com base na sobreposição de forças atrativas ou repulsivas que são aplicadas no robot.

O campo atrativo é atribuído à posição objetivo (esta representa um mínimo no espaço de configuração), enquanto que os obstáculos atuam como máximos, ou forças repelentes.

A Figura 2.5 ilustra a combinação de potencial atrativo e repulsivo.

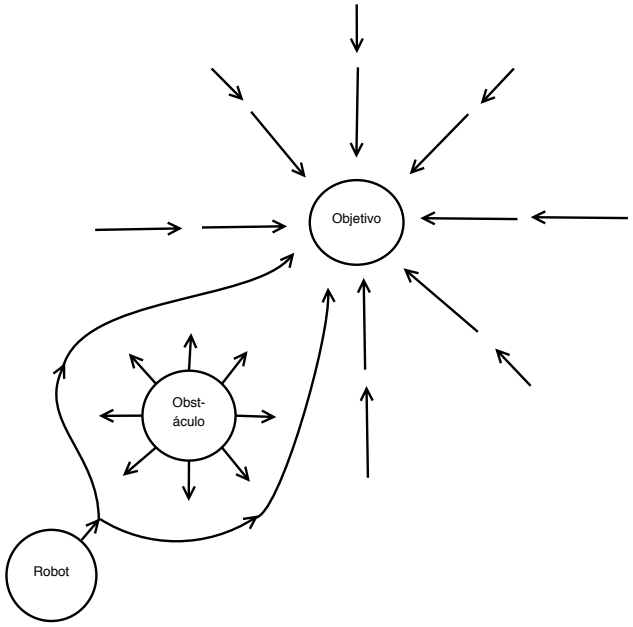


Figura 2.5: Ilustração do conceito de campo de potencial artificial.

A imagem demonstra que o *Robot* se move em direção ao seu *Objetivo* evitando ou contornando o *Obstáculo*.

Tal como se vê na figura, o *Objetivo* cria um campo potencial atrativo puxando para si o *Robot* enquanto que o *Obstáculo* cria um campo repulsivo com forças no sentido exterior, isto é, em sentido contrário. O campo potencial *U(q)* que atua no robot é então o resultado da soma do campo atrativo $U_{att}(q)$ produzido pela posição objetivo juntamente com os campos repulsivos $U_{rep}(q)$ dos obstáculos:

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{2.1}$$

Obtida esta formulação pode determinar-se a força artificial que atua no robot a qual está relacionada com o campo de potencial *U(q)* da seguinte forma:

$$F(q) = - \bigtriangledown U(q) = - \bigtriangledown U_{att}(q) - \bigtriangledown U_{rep}(q) \tag{2.2}$$

No entanto, este método tem a desvantagem de poder assumir mínimos locais da função potencial, ou seja, obter uma configuração final diferente da pretendida.

Alguns métodos foram estudados para lidar com o fenómeno do mínimo local num campo potencial. Uma opção será incorporar um planeamento de caminhos de alto-nível de modo a que o robot possa utilizar a informação que obtém do seu sensor, para corrigir o planeamento global,

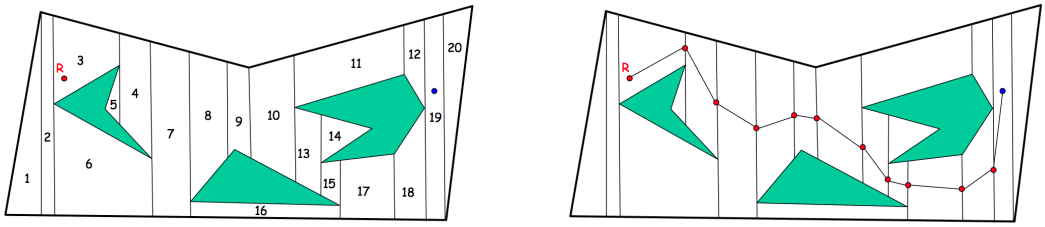
tal como fez o autor *G. Dudek* [11]. Outros métodos sugerem abordagens que permitam ao robot entrar no mínimo local e aí tentar corrigir a sua situação. Como são soluções que dependem do robot se aperceber que está bloqueado, são consideradas pouco exatas.

2.4.2 Decomposição do Espaço em Células

Existem métodos que permitem dividir o espaço livre do robot em áreas geométricas (células) livres ou ocupadas. É então criado um *grafo* da relação de adjacência entre as células, o qual vai permitir relaciona-las entre si, para depois encontrar o melhor caminho a seguir.

Os métodos baseados em decomposição de células podem ser:

- *exatos*, sempre que a união das células está contida no espaço livre (não há perdas na decomposição). O método exato mais popular é a decomposição celular trapezoidal descrita por *Latombe* [12] na qual o espaço livre do robot é varrido da esquerda para a direita desenhando uma linha vertical, paralela ao eixo dos *yy*, sempre que encontra um vértice no espaço ou obstáculos conhecidos até à fronteira do ambiente. Este método pode ser visualizado na Figura 2.6.



(a) Decomposição trapezoidal do espaço livre.

(b) Escolha de um caminho poligonal a seguir.

Figura 2.6: Ilustração da aplicação do método de forma exata da decomposição celular trapezoidal [4].

- *aproximados*, sempre que a decomposição representa uma aproximação do mapa original. A divisão celular não vai depender da forma ou tamanho dos objetos. Tem a vantagem de dar origem a sistemas mais simples de planeamento, uma vez que o custo computacional é menor, fazendo deste o método preferido nesta área.

A principal diferença entre estes métodos é o seccionamento do ambiente de trabalho poder ter ou não uma forma simples, previamente definida (levando à aproximação do mapa). Desta diferença pode inferir-se que, desde que possível, métodos exatos (ditos completos) permitem que um caminho entre duas quaisquer configurações seja obtido, desde que seja utilizado um algoritmo de busca apropriado. Por outro lado, os métodos aproximados dependem da precisão utilizada no momento de escolha da célula, o que leva a que nem sempre se encontre um caminho entre duas configurações, embora o objetivo possa ser atingido.

[4] Imagens obtidas em `http://www.ipg.pt/user/~ccarreto/_private/SR/Docs/0304/Planeamento.pdf`

Para auxiliar a compreensão da metodologia, faça-se ainda a apresentação do método trapezoidal. Com o espaço de configurações livre dividido em células e o grafo de adjacência construído, obtém-se então o planeamento do caminho em três etapas:

1. Determinam-se as células representativas da posição inicial e final do robot (ver exemplo 2.6a);

2. Faz-se uma busca no grafo para determinar o caminho de células intermédias que ligam a posição inicial à final;

3. Determina-se dentro de cada célula da solução encontrada, uma curva que ligue dois pontos entre as fronteiras dessas células (ver exemplo 2.6b).

Como referido anteriormente, tendo o ambiente decomposto, será necessário aplicar alguns métodos de busca, podendo para tal fazer-se uso daqueles que constam da secção 2.4.3.

2.4.3 Pesquisa em Grafos

Vários são os métodos conhecidos para decompor o espaço livre tendo em conta a existência de obstáculos distintos, tais como a decomposição do espaço em células - apresentada na secção 2.4.2 - diagrama de *Voronoi*, *roadmaps*, entre outros. A utilização de qualquer um destes métodos vai permitir a construção de grafos, sendo o passo seguinte encontrar um caminho que liga o nodo ou vértice inicial ao final, através de um critério de otimização mais adequado. Para isso, recorre-se aos métodos de *pesquisa em grafos* tal como refere *R. Siegwart* [9].

Nesta secção irão ser referenciados alguns dos métodos típicos de pesquisa de grafos. Pretende-se apenas fazer uma breve alusão a cada um deles, uma vez que o seu desenvolvimento está referido nos textos de alguns dos autores já mencionados para além de *S. LaValle*, tal como constam em [9, 12, 13].

Seguindo o último autor referido na frase anterior [13], em qualquer ponto da pesquisa, a exploração dos caminhos conduzirá aos nodos em um de três estados:

- **Por visitar** - Inicialmente são todos, com exceção do ponto de partida.

- **Inativos** - Nodos que já foram visitados tal como os nodos-filhos. Estão *inativos* porque não têm mais alternativas a explorar.

- **Ativos** - Nodos que foram encontrados, mas que têm, possivelmente, mais vértices para explorar. Inicialmente o único nodo *ativo* é o ponto de partida.

A partir de agora, far-se-á a apresentação de alguns dos métodos de pesquisa dos mais conhecidos.

2.4.3.1 Buscas em árvore

Há pesquisas que recorrem a algoritmos que seguem uma estrutura em árvore. Estas serão efetuadas de forma exaustiva e sistemática, percorrendo todas as possibilidades, isto significa, todos os nodos vizinhos até ao ponto de chegada, denominado raiz, de modo a que se esgotem todas as hipóteses.

Aquilo que diferencia os métodos de pesquisa em árvore entre si é a ordem pela qual os vértices são percorridos. Dado que estes apresentam uma estrutura tipo fila, as visitas aos vértices são realizadas através da ordem de chegada na estrutura (*FIFO* ou *LIFO*) eliminando algumas hipóteses. Sempre que, durante a pesquisa, se encontra um nodo inativo, avança-se para o nodo seguinte, uma vez que o mesmo não pode oferecer novas informações úteis para a pesquisa.

Pesquisa *Breadth-first* - também conhecido como pesquisa em largura ou amplitude. Como o nome indica, a expansão do algoritmo progride em largura na árvore, visitando os nodos vizinhos por ordem de proximidade ao inicial. A progressão do algoritmo é definida com o número mínimo de transição de nodos. Esta pesquisa tem a vantagem de garantir que é sempre retomado o caminho mais curto.

Uma representação deste algoritmo pode ser vista na Figura 2.7.

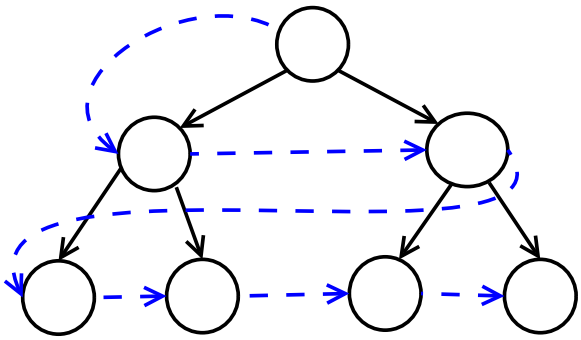


Figura 2.7: Ilustração do conceito de pesquisa em largura.

Pesquisa *Depth-first* - também conhecido como pesquisa em profundidade. Do mesmo modo, o nome indicia que a busca deste algoritmo progrida agora em profundidade, ou seja, visita todos os nodos vizinhos pela sua expansão até que estes estejam inativos.

O algoritmo está representado na Figura 2.8.

Comparando com a pesquisa *Breadth-first*, esta abordagem é considerada mais agressiva pois visa explorar os caminhos longos mais cedo, algo que é ainda destacado pelo autor *Steven M. LaValle* [13]. Isto pode ser vantajoso em termos de complexidade espacial, pois só precisa de conhecer o caminho que está a explorar. No entanto, pode ser fastidioso para grafos muito compridos.

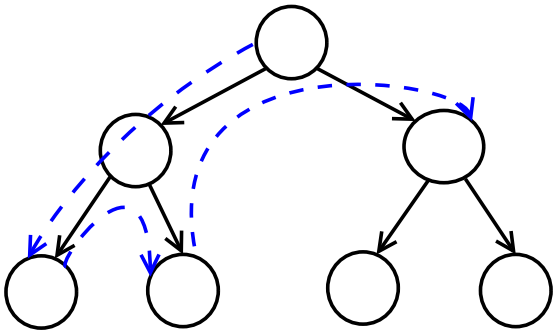


Figura 2.8: Ilustração do conceito de pesquisa em profundidade.

2.4.3.2 Buscas de custo

Os métodos de pesquisa de custo introduzem um novo elemento que é a avaliação da distância a que se encontram os novos vizinhos, permitindo, desta forma, escolher o caminho mais curto, o que o torna mais eficaz. É o caso dos exemplos que se seguem.

Algoritmo *Dijkstra* - este método tem algumas similaridades com o *Breadth-first*, garantindo também uma solução ótima. No entanto, calcula primeiro o caminho de *custo mínimo* entre todos os nós, desde o nó raiz. Posto isto, vai ver qual é o melhor caminho excluindo os nodos já visitados. É conhecido como algoritmo *guloso* porque vai selecionando o melhor caminho conforme o que lhe vai aparecendo no momento.

Para tentar obter uma melhor compreensão, apresenta-se a Figura 2.9. O nodo inicial é representado pela letra *S* e o objetivo pela letra *O*. A cinzento estão os nodos *ativos* e os destacados são os nodos vizinhos a explorar desde a nova raiz. Para isso, faz a análise de custo das arestas, a tracejado, e quando achar a pretendida, irá ser destacada tal como foi a iteração do nodo S para o nodo A na figura.

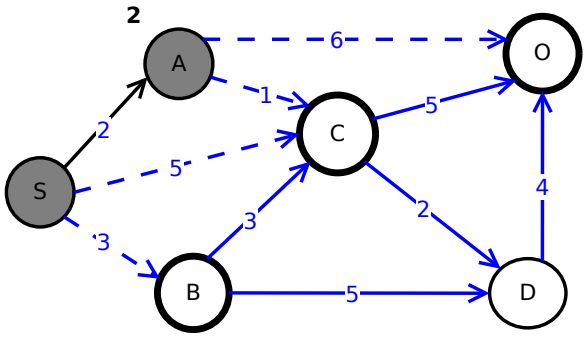


Figura 2.9: Ilustração de uma iteração do algoritmo de *Dijkstra* [5].

Os nodos só passam a *inativos* quando todas as hipóteses vizinhas estudadas foram esgotadas. No fim do algoritmo, quando o objetivo for marcado como *ativo* e todas as hipóteses vizinhas foram esgotadas, foi descoberto o caminho mais curto e pode-se identificar o retrocesso ao ponto de partida.

Algoritmo A* - Segundo *R. Siegwart* [9], este algoritmo combina aproximações heurísticas de *Dijkstra* e de *Breadth-first*. Assim sendo, tal como referido anteriormente, vai proceder à pesquisa de caminho desde o nodo inicial até ao final. Para tal, continua-se a recorrer à análise do custo (distância) esperado entre nodos - aqui conhecido por ganho g ou $g(X)$- sendo que, desta vez, faz uso para cada nodo X de uma estimativa heurística *admissível* da distância total desde esse nodo ao objetivo. Diz-se heurística admissível, representada por h ou $h(X)$, pois esta não deve ultrapassar o valor da distância ao objetivo. Logo que esse valor seja conhecido, o robot é diretamente orientado para a sua posição final, poupando estados, em comparação com o *Dijkstra*.

Assim, a função custo obtém-se computando $f = g + h$ para cada nodo e a progressão do algoritmo ocorre de modo a escolher o melhor caminho possível. Quanto menor for o custo f, maior será a sua prioridade. Em cada passo do algoritmo, esse menor valor de f é removido da fila e os valores f e g dos vizinhos são atualizados em conformidade e, então, adicionados à fila de prioridades. Constitui-se assim uma fila ordenada de segmentos de caminhos alternativos, a qual é atualizada sempre que o robot avança no seu percurso.

Uma representação deste algoritmo foi desenhada na Figura 2.10. Aqui podemos ver como a heurística ajuda o comportamento iterativo pois, quando o algoritmo não atingiu o ponto objetivo em D, retorna ao ponto anterior e assume o primeiro valor f mais baixo. Neste caso apenas o E pode ser um nodo viável. O caminho encontrado na figura em baixo foi então: S - C - D - E - O.

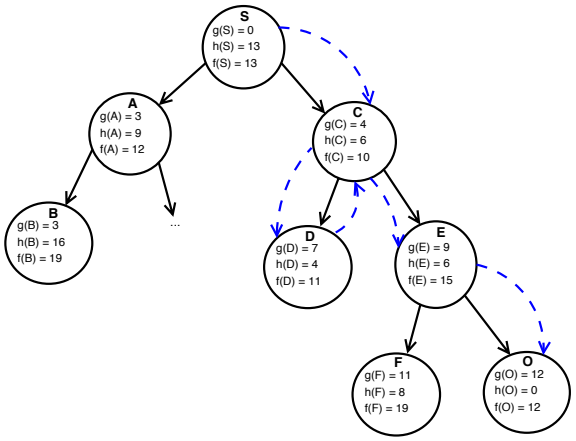


Figura 2.10: Ilustração da aplicação do algoritmo de $A*$ [6].

[5]Imagem adaptada de `http://ccnacomnet-dijkstra.blogspot.pt/`
[6]Imagem adaptada de `http://infosyncratic.nl/projects/madras/`

2.5 Problema Associado com a Não Holonomia

Um robot não holonómico é um robot que está sujeito a certas restrições nos movimentos que pode efetuar, sendo a origem destas física [14]. Se estas restrições não forem consideradas no seu planeamento, este pode resultar em movimentos que o robot não vai conseguir efetuar.

J. Barraquand [15] garante que tais restrições são geralmente causadas por um ou mais contactos de rolamento entre corpos rígidos e expressam que a velocidade relativa de dois pontos em contacto é zero. Isso faz com que a dimensão do espaço de velocidades atingível seja menor que a dimensão da configuração de espaços do robot.

Um veículo tipo carro é um robot típico não holonómico. Na ausência de obstáculos, o robot pode assumir qualquer posição no plano, com qualquer orientação. Conclui-se então que este espaço de configurações é tridimensional. No entanto, presumindo que não há deslizamento, a velocidade do ponto central das rodas traseiras é sempre tangencial à orientação do carro. Assim, o espaço de velocidades que é passível de atingir qualquer configuração é bidimensional pelos motivos referenciados no parágrafo anterior.

O planeamento de caminhos livre de colisões consiste em construir um percurso que ligue duas configurações de *input* no subconjunto de configurações, sendo que o robot não tem contacto nem interseta qualquer obstáculo. Restrições não holonómicas requerem que a tangente ao caminho, em qualquer configuração, esteja contida no subespaço das velocidades. O caminho livre de colisões para um robot não holonómico inclui tipicamente manobras [17], isto é, pontos críticos onde o robot para e inverte o sentido do movimento, o que força a uma mudança no sinal da velocidade. Nestas condições, encontrar um caminho praticável entre duas configurações é um problema complicado. Numa aplicação real é fundamental minimizar o número de manobras, ou, pelo menos, limitá-las a um número razoável.

Embora relacionadas, outras restrições serão apresentadas neste trabalho na secção 4.3.

Outros exemplos clássicos de sistemas com restrições não holonómicas são o planímetro de A. N. Krylov [16] ou corpos giratórios tais como satélites e veículos subaquáticos, para os quais a conservação do momento angular precisa ser respeitada [17].

2.6 Outras Tecnologias

Foram abordadas no decorrer deste projeto algumas tecnologias que, por algum motivo ou outro, não se adequaram às necessidades técnicas do mesmo. Atualmente, existem várias plataformas de *software* que procuram ajudar o desenvolvimento de aplicações robóticas, tais como o *ROS*, *Urbi*, *Microsoft Robotics Studio* e o *DROS*. A maioria destas estão disponíveis em código aberto, com exceção da *Microsoft Robotics Studio*, que necessita de licença comercial para utilização, sendo que todas permitem uma partilha de informação e de aplicações desenvolvidas por/para membros da comunidade robótica.

2.6.1 *ROS - Robot Operating System*

Ao contrário do que o nome sugere, o *ROS* é, mais propriamente, de um sistema meta-operativo. Este [18] fornece bibliotecas e ferramentas para ajudar os programadores de *software* a criar aplicações para robots. Assim o *ROS* oferece abstração de hardware, *drivers* de dispositivo, bibliotecas, visualizadores, transmissão de mensagens, gestão de pacotes e muito mais. Tal como foi referido, este está licenciado como fonte aberta, o que facilita o desenvolvimento de código que seja comum para a comunidade que utiliza este software.

O princípio de funcionamento do *ROS* baseia-se na troca de mensagens e serviços entre nós.

O sistema é composto por diferentes nós, cada um responsável pela sua própria tarefa. A comunicação entre estes é realizada por intermédio de mensagens, as quais são estruturas de dados comuns, do tipo inteiro, booleano, entre outros. Os serviços são definidos por um par de mensagens que correspondem a um requerimento e a sua resposta. Assim os nós publicam mensagens em tópicos que ficam disponíveis para ser chamadas por um outro nó que tenha subscrito a esse tópico.

A Figura 2.11 mostra um exemplo de uma estrutura *ROS*, *roscore*, a qual regista todos os nós, tópicos e serviços. Aqui, pode-se visualizar os nós a exercerem funções diferentes: o nó 1 apenas publica no tópico 1 o qual é subscrito pelo nó 2. Este e o nó 3, ambos publicam e subscrevem o tópico 2 e o nó 4 não comunica com nenhum dos tópicos.

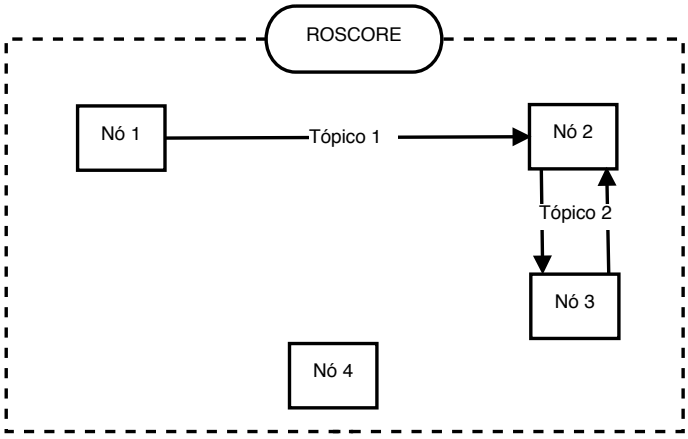


Figura 2.11: Exemplo de uma estrutura *ROS* [7].

O *Willow Garage* responsabiliza-se em disponibilizar novas versões do *ROS*. Desde a versão testada, *ROS Fuerte*, já saíram pelo menos mais duas: *Groovy* e *Hydro Medusa*. No entanto, sistemas operativos como o *OSX* continuam a não ser suportados oficialmente, tendo sido então, para a versão referida, exploradas algumas soluções para os problemas levantados nessa instalação tal como se pode ver no anexo A.

[7] Imagem adaptada de `http://www.techunited.nl/wiki/index.php?title=ROS%26OROCOS`.

2.6.2 *UWSim - UnderWater Simulation*

O *UWSim* [19] foi desenvolvido pelos investigadores do laboratório *IRSLab* na Universidade *Jaume - I* de *Castellón* - ver figura 2.12.

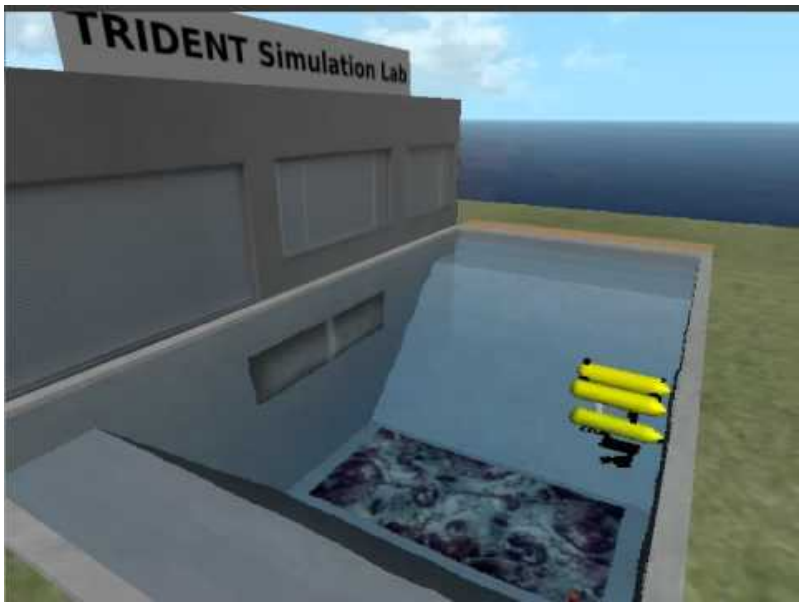


Figura 2.12: Simulação *UWSim* do projeto *Trident* da Universidade *Jaume-I* [8].

Como se pode ver na figura, este ambiente de simulação cria um cenário virtual, usando ferramentas de modelagem conhecidas, onde se pode colocar e controlar veículos aquáticos de superfície ou subaquáticos. Também é possível simular o uso de sensores que são depois acedidos externamente por interfaces de rede.

O software é implementado em *C++* e usa *OSG* (*OpenSceneGraph*) e bibliotecas de *osgOcean* para simular a renderização do meio aquático. O seu objetivo é poder fazer simulações visuais realísticas de modo a serem aplicadas numa variedade de funções como por exemplo simular missões de intervenção subaquática.

No entanto e apesar de o autor referir no site do projeto que o simulador foi testado numa variedade de sistemas operativos, em conversações com o mesmo, descobriu-se que as bibliotecas necessárias para correr o sistema não estão de facto disponíveis para todos os sistemas, o que apresenta uma maior dificuldade para quem queira usar livremente o *software*. Este facto e ainda a não disponibilização do *UWSim* de qualquer manual ou compilação de dados relevantes para quem queira usar esta plataforma torna a experiência didática muito extenuante e até pouco prática.

2.6.3 *Blender*

O *Blender* é um componente multiplataforma, que tem a vantagem de ser *open source* e que permite modelizar, compor e até mesmo animar aplicações interativas 3D.

[8] Imagem obtida em `http://i1.ytimg.com/vi/wB5zIO3e1HA/hqdefault.jpg`

Com recurso a ferramentas como dinâmica de fluidos, corpo rígido, entre outros, este programa permite a configuração do ambiente de simulação do *UWSim* desde que os ficheiros sejam mantidos num formato que o *OSG* possa ler (.ive, .3ds, .wrl, etc.).

Para o cenário básico disponibilizado pelo *UWSim*, torna-se assim possível mudar a geometria do tanque ou a definição de uma área restringida. A criação de novos nodos no *ROS* é facilitada pelo *OSG*, com uma representação virtual onde os nodos são facilmente gerados e acedidos.

2.7 Conclusões

O papel da intervenção humana na robótica móvel ainda não foi devidamente compreendido, mas este deve estar incorporado em projetos no sistema de raiz.

Portanto, não existem sistemas totalmente autónomos.

Mesmo assim, os robots vieram para ficar. As suas aplicações podem estar genericamente relacionadas com questões de transporte, vigilância e exploração. Para tal, a problemática envolve várias ramificações como as limitações de não colisão, a identificação e a sua localização no ambiente, assim como o planeamento de movimento.

Este último será o objetivo fulcral deste trabalho durante o qual se pode fazer recurso a algoritmos de pesquisa para obter o caminho que o robot pode escolher, evitando obstáculos e desvios nada práticos.

Existem já disponíveis muitas ferramentas para analisar, visualizar e tratar dados de missões em meio marítimo mas a simulação ainda é um assunto delicado devido às muitas variáveis a controlar, tanto física, como computacionalmente. Assim, desde que seja viável uma representação simples, pode-se visualizar todo o movimento do robot ideal através de um programa direto de simulação como é o MatLab® e tenta-se computar erros.

Capítulo 3

Descrição do Problema

No presente capítulo aborda-se a problemática do planeamento e execução de caminho de um veículo autónomo aquático. Considerando que não existem outras restrições, são obtidos caminhos de comprimento mínimo através da concatenação de curvas, com raio R, e retas ($R \rightarrow \infty$).

A solução vai depender da pose inicial e final, assim como limitações da orientação. Partindo deste pressuposto, para obter uma deslocação eficiente do robot de acordo com os objetivos e requisitos especificados pelo utilizador ou por um nível de organização superior, é necessário ter em conta as restrições impostas pelo ambiente e pelo veículo.

3.1 Modelo do Movimento

Os modelos matemáticos são utilizados para representar o comportamento físico do robot, por forma a se poder organizar e depois controlar o movimento. Deste modo, os modelos matemáticos têm de representar evolução do estado do sistema ao longo do tempo, consoante os controlos aplicados ao robot.

Existem modelos cinemáticos e dinâmicos, sendo que os primeiros são os mais simples e tratáveis. Estes modelos baseiam-se sobretudo em questões geométricas da mecânica do robot. Os estados considerados nestes modelos são normalmente posição e orientação. No caso dos modelos dinâmicos considera-se também as velocidades angular e linear do veículo.

Por forma a se conseguir utilizar estes modelos, é normal assumirem-se algumas simplificações, sendo que estas vão depender da aplicação.

Quando não se considera a existência de obstáculos, então não é necessário escolher a forma do robot, uma vez que este não pode colidir. No entanto, quando estes existem, a forma do robot vai afetar o modo como os obstáculos impõem restrições nas configurações admissíveis. Assim, para um dado obstáculo e uma dada forma de robot, podemos obter uma restrição nas configurações de forma:

$$F(x, y, \theta) > 0 \tag{3.1}$$

No entanto, quando o robot é considerado como um ponto, a restrição não vai depender da orientação.

De seguida, faz-se uma apresentação do sistema com recurso a equações diferenciais, tomando em consideração as limitações do veículo.

O AUV MARES

A dissertação foi desenvolvida tendo por base um veículo subaquático autónomo (AUV) MARES presente no laboratório do grupo *OceanSys*, da Faculdade de Engenharia da Universidade do Porto, onde foi desenvolvido.

De acordo com o *site* do projeto [20], o AUV MARES tem a característica de ser altamente modular. São várias as secções em que está organizado. Tem por isso a vantagem de facilmente se poder fazer trocas de peças permitindo carregar sensores *on-board* segundo a missão que vai executar. A existência dos diferentes sensores é bastante útil para adquirir informação durante a localização e navegação do mesmo.

A maior particularidade do veículo, que pode ser visualizado na Figura 3.1, é o sistema de propulsão, totalmente desacoplado, que é composto por um par de propulsores verticais e outro horizontal. Isto permite ao MARES fazer movimentações na horizontal ou na vertical de forma distinta, permitindo-lhe circular em áreas confinadas como no fundo de um contentor de água ou executar inspeções muito próximas do objeto ou superfície a acostar.

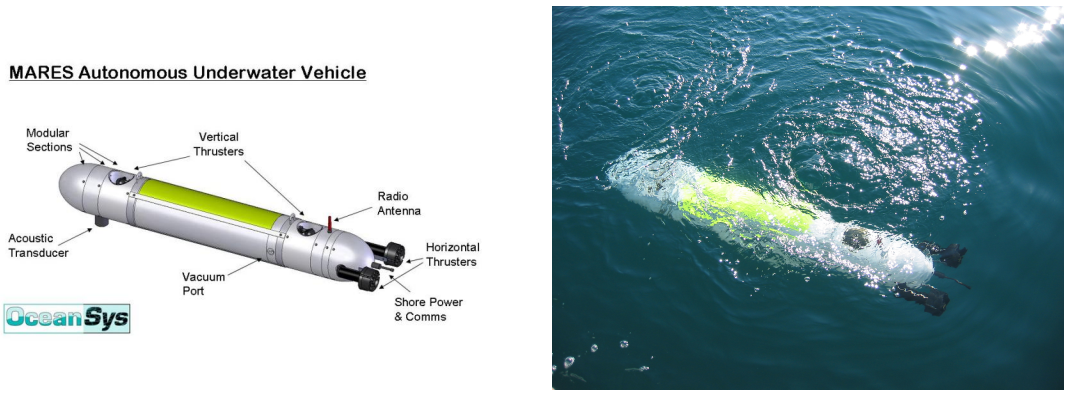


Figura 3.1: Ilustração do AUV MARES e uma imagem do mesmo pronto para a missão [1].

Outra característica do MARES relevante é o comprimento de 1.5 *m* e 20 *cm* de diâmetro que lhe permite atingir uma velocidade de deslocamento horizontal máxima de 2 *m/s*.

Considere-se, então, daqui para a frente, que o veículo autónomo utilizado é muito mais comprido do que largo, sendo por isso mesmo o ponto de referência do seu movimento o centro de massa. Se o veículo se deslocar em linha reta, este ponto move-se ao longo da mesma. Se o caminho for curvilíneo, o deslocamento deste ponto é feito ao longo de uma circunferência de raio R,

[1] Esquemático obtido em [20]

limitado pelas restrições físicas do veículo (por exemplo: dimensões, forma,...). Neste tipo de trajetória, o movimento do veículo, em cada ponto, é caracterizado então pelo raio da circunferência e pelo ângulo de viragem θ.

Assim, para representar o veículo, as coordenadas do ponto central definem uma configuração $q_p = (x_p, y_p, \theta_p)$ onde (x_p, y_p) representa a posição do veículo no instante p e θ_p a orientação do mesmo.

Para um intervalo de tempo muito pequeno, é comum ter $\frac{dy_p}{dx_p} = \tan\theta_p$.

Como $\frac{dy_p}{dx_p} = \frac{\dot{y}_p}{\dot{x}_p}$ e $\tan\theta_p = \frac{\sin\theta_p}{\cos\theta_p}$, pode-se escrever a equação de restrição:

$$-\dot{x}_p \sin\theta_p + \dot{y}_p \cos\theta_p = 0 \tag{3.2}$$

Reescrevendo esta restrição, facilmente se chega às equações diferenciais de pose do modelo:

$$\begin{aligned} \dot{x}_p &= v_P \cos\theta_p \\ \dot{y}_p &= v_P \sin\theta_p \end{aligned} \tag{3.3}$$

Quando o robot se desloca em torno de um círculo de raio R_{curva}, sendo v_p a velocidade a que este se desloca, irá ter uma velocidade angular de:

$$\dot{\theta} = \omega = \frac{v_p}{R_{curva}} \tag{3.4}$$

Se o ângulo de viragem está fixo, então a curvatura será máxima $\left(k = \frac{1}{R_{curva}}\right)$. A curvatura é feita em torno do centro de rotação. Para a curvatura máxima k, o raio R_{curva} é mínimo, sendo representável por R_{min}.

Pretende-se que o caminho apropriado para o veículo seja definido por um conjunto de configurações que ligue uma configuração inicial, q_i, até uma configuração final q_f, respeitando a equação de restrição (3.2).

Numa primeira fase, o meio em que o veículo se irá deslocar é estático, o que significa que não será necessário ter em consideração correntes marítimas.

A variável independente do modelo, tendo em conta o meio estático e as restrições do veículo, é a distância percorrida, representada por s.

Pode-se reescrever o modelo cinemático em função desta variável da seguinte forma:

$$\begin{cases} \dot{x}_p = v_P \cos\theta_p \\ \dot{y}_p = v_P \sin\theta_p \\ \dot{\theta} = \frac{v_p}{R_{curva}} \end{cases} \tag{3.5}$$

Na realidade, o tipo de robot considerado é um sistema não linear o que quer dizer que o sistema apresentado em (3.5) é, no fundo, apenas uma aproximação do sistema real. Sendo assim, torna-se necessário atualizar estas equações para que embora ainda aproximado, o sistema seja o mais fiel possível ao real, isto é, para cada iteração pretende-se que o robot vá atualizando a posição. Daí resulta que:

$$\begin{aligned} x_{atual} &= x_{anterior} + \delta v_p * cos(\theta) \\ y_{atual} &= y_{anterior} + \delta v_p * sin(\theta) \\ \theta_{atual} &= \theta_{anterior} + \frac{\delta v_p}{R_{curva}} \end{aligned} \tag{3.6}$$

O símbolo δv_P representa a amostragem que se faz entre dois pontos do percurso da deslocação do veículo, atualizando x_atual para o total de pontos do veículo. Se δ for muito pequeno, então os pontos estão perto um do outro.

Para que esta deslocação se aproxime daquela que é feita na realidade, são necessários um número de pontos que permita definir com precisão essa movimentação. Assim, quanto maior for o número de pontos definidos, melhor se visualizará o percurso a efetuar pelo veículo.

3.2 Composição do Movimento

O modelo cinemático considerado foi introduzido por *L. Dubins* em 1957 [21]. Este matemático americano descreveu e apresentou uma solução para o problema de caracterização do caminho mais curto que uma partícula poderá descrever num plano, com uma velocidade linear constante.

Este modelo aplica-se em (3.6). Modificando a velocidade v_p para a velocidade unitária do carro de *Dubins*, a deslocação adquire um único sentido.

Mais tarde, os autores *J. Reeds* e *L. Shepp* [22] continuaram a desenvolver o mesmo problema, introduzindo adicionalmente o movimento com sentido inverso ou à retaguarda, usando velocidades negativas, com o auxílio da variável de controlo da velocidade, $|v| \geq 1$.

Na sequência do que foi mencionado atrás, robots tipo carro (ou como os mencionados noutras secções como 2.3 e 2.5) têm como elemento comum o raio de curvatura mínimo. Este é como um circulo tangencial ao veículo que, por muito que este tente, só poderá seguir a linha da circunferência com R_{min}, mesmo que o veículo faça uma viragem mais apertada.

Assim, seguindo estes autores e admitindo que não existem outras restrições, são disponibilizados métodos que fazem o uso de manobras elementares para decompor os movimentos de um veículo autónomo. Assim, caminhos de comprimento mínimo para o modelo cinemático podem ser obtidos pela concatenação de curvas, de raio conhecido R, definindo arcos de circunferência e segmentos de linha reta. A solução exata vai depender da posição relativa inicial e da pose final e, também, das limitações na orientação final.

Finalmente, parte-se para um movimento chamado composto, o qual congrega a soma das manobras elementares mais simples sendo estas o movimento de translação lateral e a rotação do veículo sobre o seu próprio eixo, as quais serão apresentadas no próximo capítulo 4.

3.3 Conclusões

Neste capítulo foram abordadas as questões relativas à movimentação de um robot, sendo que estas podem ser mais ou menos exatas consoante o número de pontos definidos para sua trajetória. A história revela que esta problemática tem sido estudada ao longo do tempo e evoluído continuamente. Assim, inicialmente, os movimentos considerados partiam de sistemas dinâmicos lineares para definir uma aproximação aos sistemas não lineares como os do carro de *Dubins* ou os de um robot. A introdução de novas variáveis permitiu a definição de movimentos mais complexos e mais próximos dos reais executados pelo robot durante o seu percurso.

As equações do modelo obtidas permitem, através das velocidades de controlo, fazer a simulação do caminho muito realista que o robot irá percorrer, caso não surjam limitações no robot ou no meio ambiente.

Capítulo 4

Manobras Elementares

Nesta secção abordar-se-á a problemática da definição de manobras possíveis para robots, tendo em conta o espaço livre dentro do espaço de configuração.

A situação que aqui se apresenta desenvolve-se em duas variantes: uma mais simples, que tem o objetivo de ilustrar a deslocação do veículo autónomo afetada apenas pelas restrições sobre o movimento do mesmo e, outra, no caso de existirem obstáculos no meio em que o robot se desloca as quais interferem com o seu movimento.

Para além de se referir o trabalho prévio necessário para uma experiência laboratorial, também haverá referências ao assunto, a partir da apresentação feita no artigo [23]. Esta foi o ponto de partida para a estruturação deste capítulo.

4.1 Translação Lateral

Para a manobra de translação lateral, definiu-se que o melhor caminho possível é composto por dois arcos de circunferência em concordância reversa e por um segmento de reta paralelo ao eixo dos *xx* que faz a ligação entre o segundo arco de circunferência e o eixo dos *yy* na direção deste. Através desta sequência, o veículo desloca-se $|y_0|$ segundo o eixo dos *yy* e mantém a sua orientação e posição segundo o eixo dos *xx*, como pretendido.

Convém recordar que uma configuração $q = (x, y, \theta)$ é uma especificação de localização, onde (x, y) representam a posição do robot e θ a orientação do mesmo. O eixo foi estipulado como pode ser visto na Figura 4.1. Posto isto, o referencial considerado deverá abranger a deslocação tanto no sentido positivo (a alteração de configurações $(0, |y_0|, 0)$ para $(0, 0, 0)$ na origem) como no sentido negativo, ou seja, com incremento negativo em *yy*.

4.1.1 Determinação dos Parâmetros

Como referido anteriormente, a manobra pode ser dividida em três segmentos que ligam os pontos P_0, P_1, P_2 e P_3 (ver Figura 4.1). Durante a sua execução foi considerada a velocidade constante em módulo para cada segmento definido pelos referidos pontos. Estes vetores velocidade deverão respeitar as restrições não holonómicas presentes em (3.2) e são tangentes à trajetória.

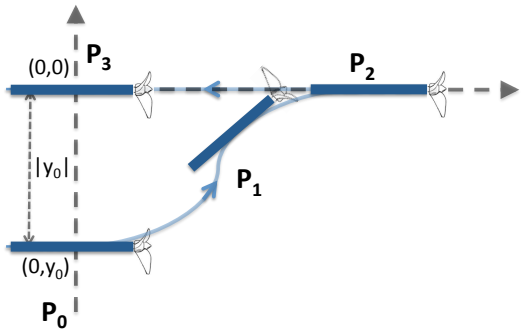


Figura 4.1: Manobra de translação lateral em sentido positivo.

Para as duas curvas tem-se no instante p:

$$V_1 = \begin{bmatrix} v_p \times \cos\theta_p \\ v_p \times \sin\theta_p \\ \frac{v_p}{R_1} \end{bmatrix}, V_2 = \begin{bmatrix} v_p \times \cos\theta_p \\ v_p \times \sin\theta_p \\ \frac{v_p}{R_2} \end{bmatrix} \tag{4.1}$$

em que se percorre o arco de circunferência desde P_0 a P_1, com um raio de curvatura $|R_1| = cte \geq \rho_{min}$. O arco concorrente de P_1 a P_2 tem o mesmo raio de curvatura, mas perfaz o sentido inverso $|R_2| = cte \geq -\rho_{min}$. Assim, para este caso, R_2 é o mesmo que $R_2 = -R_1 = |R|$.

O segmento de reta, distância percorrida entre P_2 e P_3, pode representar a velocidade apenas no eixo dos xx, ou seja, do seguinte modo:

$$V_3 = \begin{bmatrix} v_p & 0 & 0 \end{bmatrix}^T \tag{4.2}$$

Definidos os vetores velocidade em cada segmento, passa-se então à determinação dos parâmetros de deslocamento, isto é, o comprimento dos segmentos a percorrer:

$$\begin{aligned} &s_1 = s_2 = R \times \theta_1; \\ &s_P = -[R_1 \sin\theta_1 + (-R_2)\sin\theta_1] = 2R \times \sin\theta_1 \end{aligned} \tag{4.3}$$

Com o auxílio da Figura 4.2, pode-se facilmente determinar $|y_0| = R - R\cos\theta_1$. Neste caso específico, obtém-se:

$$|y_0| = (R_1 - R_2) \times (1 - \cos\theta_1) = 2R \times (1 - \cos\theta_1) \tag{4.4}$$

Por fim, a distância conhecida $|y_0|$ permite conhecer o ângulo percorrido na durante a duração dos arcos:

$$\theta_1 = \arccos\left(1 - \frac{|y_0|}{2R}\right) \tag{4.5}$$

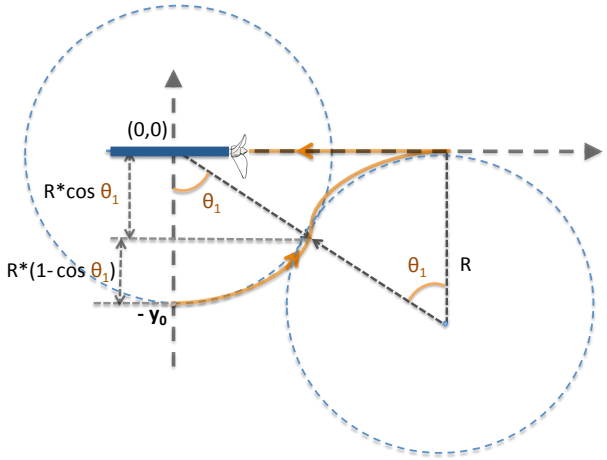


Figura 4.2: Determinação do ângulo percorrido na duração dos arcos.

4.2 Rotação Pura

Quando se fala em rotação pura, convém recordar que o veículo não consegue rodar diretamente sobre si próprio. A ação diferencial dos propulsores do veículo autónomo provocam um movimento curvilíneo, em torno do raio de curvatura, ρ_{min}.

A rotação pura consiste então na alteração da orientação do veículo mantendo a sua posição cartesiana final. Observando a estipulação de eixos considerada em 4.1, a manobra da rotação vai consistir em mover o veículo desde a configuração $(0,0,0)$ para $(0,0,\theta_f)$ (expressado na forma $q = (x,y,\theta)$), como pode ser visto na Figura 4.3 .

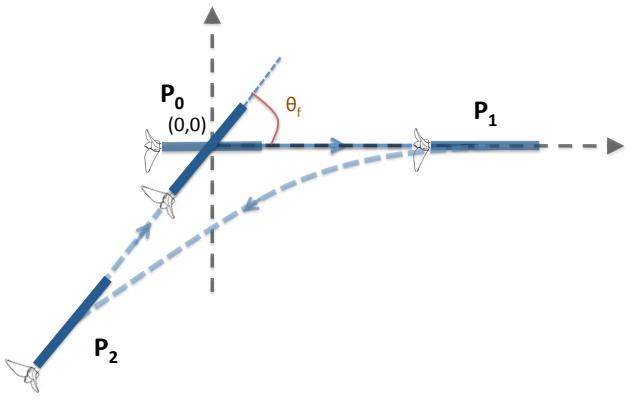


Figura 4.3: Manobra de rotação no sentido positivo.

4.2.1 Determinação de Parâmetros da Rotação

Os vetores de velocidade da rotação serão iguais a V_1 e V_2, referidos em (4.1), podendo usar-se qualquer um como referência. No entanto, o movimento de translação apresenta agora uma velocidade ao instante p:

$$V_P = \begin{bmatrix} v_p \cos\theta_p & v_p \sin\theta_p & 0 \end{bmatrix}^T \tag{4.6}$$

Os comprimentos dos segmentos que constituem a manobra de rotação representada na Figura 4.3 são obtidos através das seguintes expressões:

$$\begin{aligned} s_2 &= R \times \theta_f; \\ s_{P1} = s_{P2} &= \frac{R \times (1 - \cos\theta_f)}{\sin\theta_f} \end{aligned} \tag{4.7}$$

Na figura referida, o veículo descreve inicialmente, um segmento de reta representado em (4.7) por s_{P1} de P_0 a P_1. Depois tem-se um arco de circunferência com raio R (de P_1 a P_2), e finalmente, volta a percorrer o segmento de reta de P_2 a P_0, representado por s_{P2}. Tendo s_{P1} e s_{P2} o mesmo comprimento e considerando o raio mínimo de curvatura, o valor de deslocamento mínimo do veículo verifica-se para $|R| \geq \rho_{min}$.

4.3 Restrições

Nesta secção, aborda-se o tema de planeamento do movimento do robot em ambientes onde existem limitações físicas do espaço de configuração. A abordagem seguida baseia-se na resolução do problema que é o estacionamento de um veículo autónomo.

O número de manobras total que o veículo terá que fazer vai depender de vários fatores. Já foram referidas as restrições não holonómicas. Agora, convém considerar a forma do robot e o espaço físico de movimentação, sabendo que quanto menor este for, maior será o número de manobras.

4.3.1 Afastamento do Veículo

Para garantir a integridade do veículo autónomo é necessário calcular a projeção da envergadura deste ao longo do caminho determinado.

Para tal, utiliza-se como referência o veículo numa posição da manobra com uma dada inclinação θ_1. Há que ter ainda em conta o comprimento do veículo L, portanto $\frac{L}{2}$ para cada segmento desde o centro de massa, como se pode observar na Figura 4.4.

O afastamento do veículo (x_a, y_a) em relação à origem, do sistema de coordenadas considerado, tal como quando realiza a manobra com a concavidade invertida, é obtido no ponto (x_P, y_P), em módulo:

$$\begin{aligned} |x_a| &= x_P + \frac{L}{2} \times \cos\theta_1 \\ |y_a| &= y_P + \frac{L}{2} \times \sin\theta_1 \end{aligned} \tag{4.8}$$

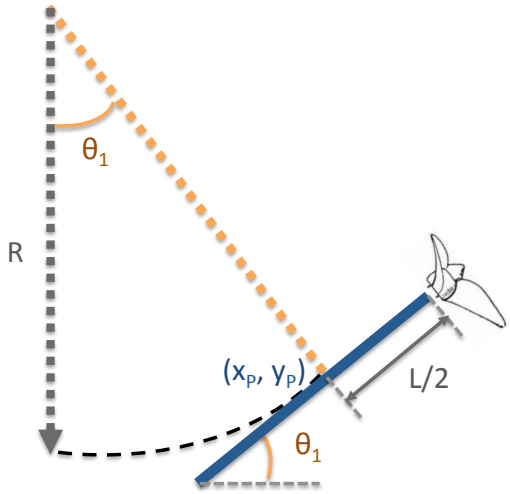


Figura 4.4: Determinação do afastamento do veículo.

4.3.2 Espaço Disponível

O espaço funciona como uma condicionante para o movimento do robot. De seguida, são esboçadas as respostas que se espera que o veículo dê para o caso de limitação do espaço que este necessita para desenvolver as suas manobras típicas de deslocação.

No caso de um espaço longitudinal, no qual o veículo tem que se mover, o número total de manobras que este terá que fazer, vai depender do espaço disponível. Desta forma, se o espaço for muito limitado, aumentará o número de manobras.

O espaço disponível é uma variável a considerar também para o caso da profundidade, a qual se apresenta como uma restrição pois pode ser alargado ou diminuto. Este aspeto não chega, porém, a ser abordado. No entanto, sempre que o espaço seja diminuto terá de se prever o aumento do número de manobras necessárias para proceder à imobilização do veículo autónomo tal como acontece na situação descrita no parágrafo anterior.

4.3.2.1 Translação Lateral

Com a atualização da pose, pretende-se fazer, numa primeira instância, com que o veículo autónomo percorra uma dada distância lateral, cujo espaço disponível está restrito na horizontal.

Para tal, como foi visto em (4.5), é possível obter, a partir de $|y_0|$, a orientação necessária, θ_{nec}, para o veículo executar a manobra de translação lateral.

Aplicando esta orientação nos parâmetros da translação lateral presentes em (4.3), pode-se verificar se o espaço disponível, s_P_disp, é o suficiente para fazer toda a manobra de uma só vez dentro do espaço restrito, ou seja, se s_P_nec é menor ou igual a s_P_disp. Se assim for, pode realizar-se a manobra. Caso isso não aconteça, o θ_{nec} é atualizado para o θ_{atual} que o veículo consegue percorrer, diminuindo sucessivamente a distância $|y_0|$ que se atualiza sempre que há uma manobra tornando-se $|y_0|_{atual}$ que falta para atingir o seu objetivo. Neste caso a translação lateral concretiza-se da mesma forma, mas por manobras sucessivas.

O algoritmo foi implementado como se apresenta em 1.

Algoritmo 1 Atualização de pose para a translação lateral

Require: $y_0, R, s_P\_disp \neq 0$

while $y_0 > 0$ **do**
  $\theta_{nec} = \arccos\left(1 - \frac{|y_0|}{2R}\right)$
  $s_P\_nec = 2R \times \sin(\theta_{nec})$
  if $s_P\_nec < s_P\_disp$ **then**
    realizar manobra
  else
    atualizar $\theta_{nec} \leftarrow \theta_{atual} = \arcsin\left(\frac{s_P\_disp}{2R}\right)$
    atualizar $y_0 \leftarrow y_0\_atual$
    realizar a manobra com θ_{atual} e y_0_atual
  end if
  calcular o y_0 que falta percorrer
end while

4.3.2.2 Durante a Rotação Pura

Definida a translação lateral, procede-se à mudança de orientação do veículo, recorrendo-se a uma manobra de rotação pura. Neste caso definiu-se a orientação inicial como sendo $\theta_{atual} = 0$. Contudo, este valor não é restritivo.

Ao fazer esta rotação pura, pretende-se obter a orientação final do veículo autónomo, a qual é definida pelo utilizador, sendo que a restrição do espaço disponível permanecerá na horizontal, isto é, mantém-se o mesmo tipo de restrição que a analisada no ponto anterior.

De modo semelhante, a pose irá ser atualizada com recurso aos parâmetros do deslocamento apresentados em (4.7).

A partir da orientação pretendida, θ_{falta}, verifica-se o espaço disponível, s_P_disp, para realizar a manobra de rotação, tal como foi feito anteriormente. Do mesmo modo, se o espaço necessário exceder esta restrição, ou seja, o s_P_nec for maior que s_P_disp, procede-se à atualização da pose: percorre-se a orientação possível, θ_{disp}, realizando então o número de manobras necessário até que a diferença para θ_{falta} seja nula.

O algoritmo 2, apresentado de seguida, representa um sumário da implementação dos passos necessários para a realização da manobra.

Algoritmo 2 Atualização de pose para a rotação

Require: $\theta_{falta}, R, s_P\_disp \neq 0$
Ensure: $\theta_{atual} = 0$
1: **while** $\theta_{falta} > 0$ **do**
2: $s_P\_nec = R \times \frac{1-\cos(\theta_{falta})}{\sin(\theta_{falta})}$
3: **if** $s_P\_nec < s_P\_disp$ **then**
4: realizar a manobra
5: **else**
6: $\theta_{disp} = 2\arccos \times \frac{R}{\sqrt{s_P\_disp^2 + R^2}}$
7: atualizar $\theta_{falta} - \theta_{disp}$
8: realizar a manobra para θ_{atual}
9: **end if**
10: calcular o θ_{falta} que falta percorrer
11: **end while**

4.4 Conclusões

O espaço disponível para a manobra de um robot condiciona os movimentos a realizar, pelo que a existência de obstáculos ou barreiras são elementos a ter em conta no planeamento do movimento.

O tipo de movimento a escolher está quartado pelas condições naturais em que o veículo autónomo se desloca.

A escolha dependerá sempre da análise que, no momento, for feita pelo algoritmo das variáveis encontradas ao longo do trajeto.

Capítulo 5

Simulação

Como primeiro passo para testar a viabilidade dos métodos, é necessário efetuar simulações que ponham em prática a caraterização do movimento. A simulação decorreu sobretudo no ambiente MatLab® e será apresentada de seguida com uma breve descrição do que foi obtido. O veículo autónomo foi, neste caso, representado como uma partícula pontual que pretende deslocar-se em espaço livre ou limitado. As restrições do movimento foram introduzidas diretamente no programa.

Em cada gráfico, o movimento do robot, apesar de contínuo, apresenta-se discretizado de modo a se compreender melhor os passos dados. Da mesma maneira, o movimento mais elementar é identificável pelo uso de cor, de forma a diferenciar os três estados (do movimento).

Nos vários resultados que vão ser referidos nesta secção, estes assumem que a curvatura admissível para a representação do veículo autónomo é dada por um raio de curvatura unitário. Tal hipótese não é restritiva, uma vez que, caso um robot tenha outras dimensões, através da mudança de escalas, o movimento pode ser tornado equivalente ao do robot com raio de curvatura unitário, podendo ser maior ou menor de acordo com o seu tamanho.

5.1 Deslocamento Simples

Represente-se um robot virtual de referência num determinado ponto, o qual se move ao longo do tempo.

Tipicamente, se se fosse acostar um veículo, antes de começar a manobra, a posição e orientação deste têm que ser ajustadas à localização da acostagem. Do mesmo modo, admite-se que a pose inicial é conhecida. A manobra é então executada iterativamente com moções de movimento lentas (tipo frente-marcha atrás).

Este tipo de deslocação diz-se *simples* por se ter um espaço de configuração completamente livre, o que permite ao robot fazer o deslocamento no sentido ou direção que se quiser, sem haver preocupações com possíveis colisões. Tal como referido na secção 4.1 e 4.2, os eixos são estipulados de modo a que o robot se mova livremente em torno do ponto central.

Na Figura 5.1 é possível observar o ponto a efetuar uma manobra de translação lateral simples, o que quer dizer que este vai efetuar uma variação total de movimento no eixo dos yy. O veículo autónomo deverá efetuar uma manobra que lhe permita terminar paralelamente à posição inicial, ou seja, com a mesma orientação inicial.

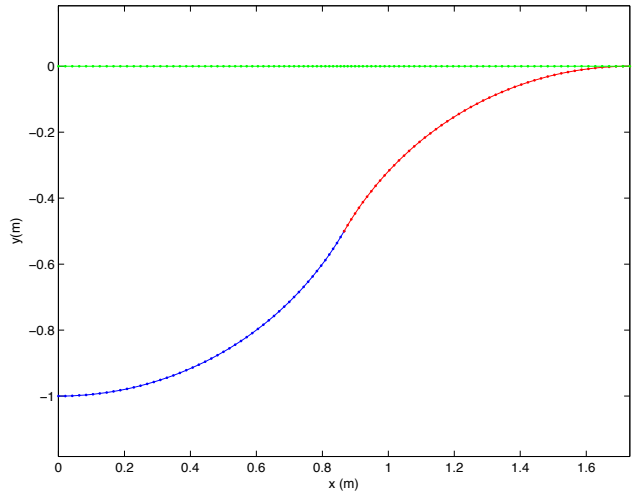


Figura 5.1: Execução da manobra de translação lateral simples, ou seja, sem qualquer restrição, para $\triangle\, y = 1\text{m}$.

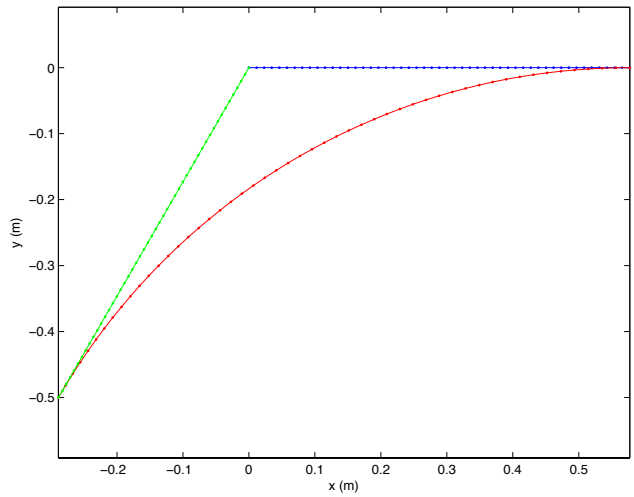


Figura 5.2: Execução da manobra de deslocamento rotacional puro para $\triangle\, \theta = 60^{\circ}$.

Para isso, o veículo parte do ponto inicial $(0,0,0)$, efetua uma rotação da posição em direção ao objetivo, durante metade do percurso, a qual está representada na figura com a cor azul, efetuando depois a correção da trajetória com outra rotação, a vermelho na figura, de modo a chegar a um

ponto na linha do objetivo. Assim que alinha com este, executa uma reta linear, que se pode ver marcada a verde, até chegar à posição desejada em $(0,1,0)$.

Já na Figura 5.2, o ponto efetua uma manobra de rotação pura, ou seja, uma variação de orientação, a qual é especificada pelo utilizador.

Para uma deslocação de 60°, o ponto, ao partir da origem em $(0,0,0)$ faz um deslocamento, a azul, para que tenha espaço suficiente para percorrer uma curva, a vermelho na figura, que vai permitir fazer a correção da orientação final para a pretendida. A verde está o movimento de retorno à posição desejada, que é o ponto inicial, já com a orientação final.

5.2 Deslocamento Múltiplo

De seguida, é apresentado o mesmo movimento mas com restrições no espaço. Aqui o veículo continua a fazer movimentos iterativos durante o deslocamento até que esteja na pose final pretendida, seja esta o estado paralelo ou a orientação desejada. O número de manobras total que o veículo terá que fazer vai depender do espaço longitudinal disponível.

Neste caso, apresenta-se na Figura 5.3 um movimento de translação lateral, para uma distância de 1 m, em que o veículo foi sujeito a uma restrição de espaço horizontal, isto é, foi-lhe fornecida a informação do espaço disponível longitudinalmente.

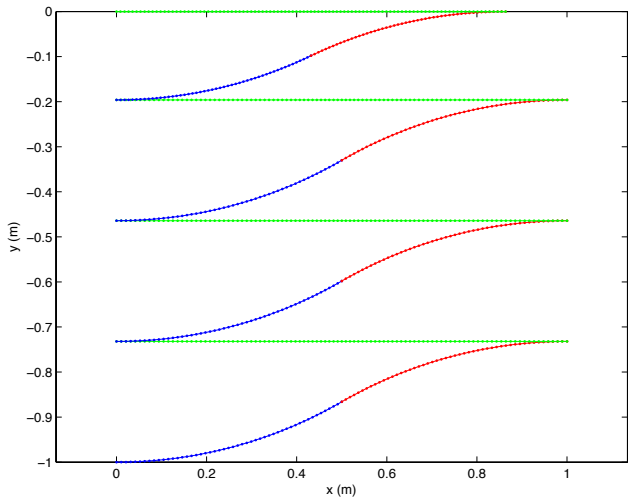


Figura 5.3: Efeito da manobra de translação lateral múltipla para $\triangle y = 1m$ com restrição de espaço disponível horizontal $x = 0.9$m.

Atendendo ao espaço a percorrer, foi necessário efetuar várias manobras até chegar ao objetivo pretendido. Isto acontece porque não é possível com uma única manobra atingir o objetivo definido. Para estas aproximações sucessivas foi utilizado o algoritmo previamente delineado em 1. Durante o percurso, o veículo vai atualizando o θ_{nec} para poder fazer cada uma das curvas, fazendo o mesmo para a distância que falta até ao objetivo final.

Assim, a deslocação do veículo é feita por aproximações sucessivas com a atualização constante dos dados de que necessita para proceder à sua movimentação.

Quanto ao movimento de rotação múltiplo, a restrição do espaço longitudinal mantém-se. Iterativamente, a disponibilidade do espaço é avaliada, recalculando o valor da orientação possível, θ_{disp}, até que a posição desejada seja atingida.

Na Figura 5.4 mostra-se a simulação deste cálculo. Para o deslocamento de 90°, o espaço disponível é calculado, mas, como não é possível atingi-lo com uma só manobra, o veículo terá de ajustar a sua posição. Neste caso, a manobra é feita em duas fases distintas. Em cada iteração do movimento, a orientação vai ser diferente de acordo com o algoritmo 2, pelo que o ponto faz um trajeto no sentido de evidenciar essa deslocação.

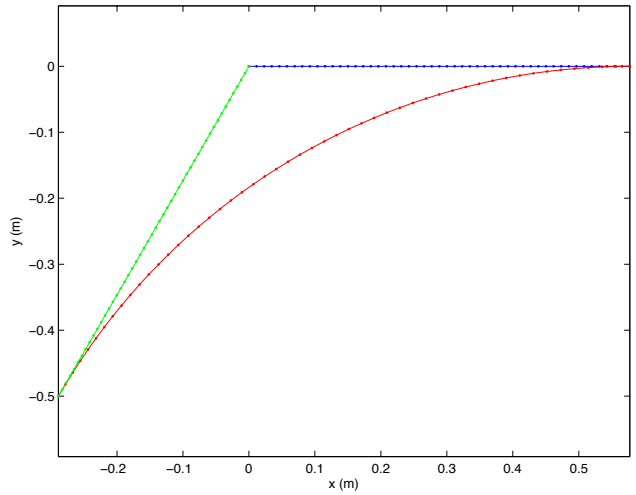


Figura 5.4: Ilustração da manobra de rotação múltipla para $\triangle\ \theta = 90°$com restrição de espaço disponível horizontal $x = 0.5m$.

Em qualquer uma das manobras, o *script* desenvolvido realiza a contagem do número necessário destas para efetuar o deslocamento total que foi pedido, no caso da Figura 5.3 são 4, e no da rotação, na Figura 5.4, 2.

5.3 Área Envolvente

O robot entra num espaço ou área restrita e move-se ao longo das suas fronteiras. Para garantir uma área de segurança dentro do espaço de configuração, foram desenhadas áreas envolventes das manobras elementares ao longo da duração das mesmas. Isto pode permitir, por exemplo, garantir uma zona de segurança, durante a qual o veículo não possa intersectar possíveis obstáculos presentes no espaço.

Esta zona de conforto é importante para a segurança do veículo, quer durante uma manobra simples, quer se se tratar de uma manobra múltipla. Pode-se observar na Figura 5.5 a deslocação do veículo na lateral, sempre dentro dum espaço em que não se vislumbram obstáculos que possam impedir a trajetória ou danificar o veículo. Visto que se trata de uma deslocação paralela faz sentido que esta manobra se desenvolva ao longo de um espaço retangular.

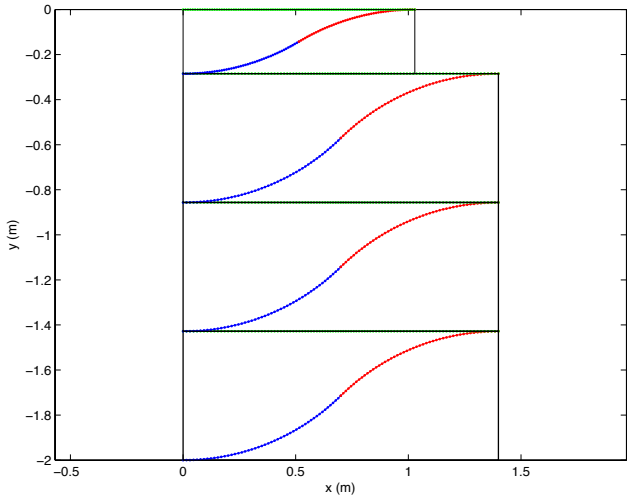


Figura 5.5: Ilustração como em 5.3 com área envolvente por manobra elementar.

Do mesmo modo, sempre que o veículo se desloca em rotação pura, o espaço a percorrer tem uma área envolvente alargada representada na Figura 5.6 pelo círculo exterior o qual deverá permanecer livre para que o total de manobras a efetuar pelo veículo possa decorrer com normalidade. Sempre que isto acontece, está-se a assegurar a possibilidade de realização de uma segunda manobra, representada na figura pelo círculo interior, igualmente sem problemas de colisão.

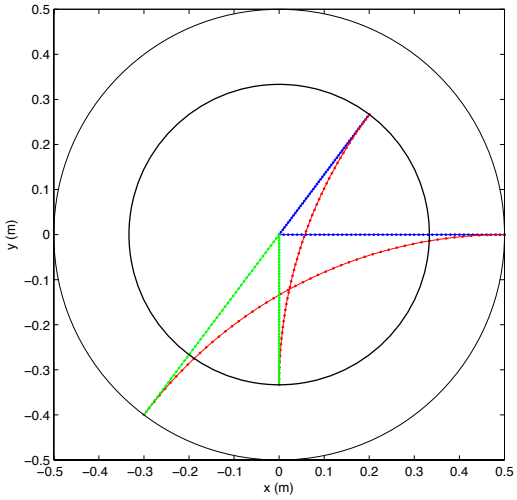


Figura 5.6: Ilustração como em 5.4 com área envolvente por manobra elementar.

5.4 Concatenação das Manobras

De modo a poder realizar as manobras em conjunto, executando uma de cada vez, desenhando um caminho, é necessário fazer um trabalho de implementação mais profundo.

Partindo do acesso aos pontos de partida, pois estes podem ser conhecidos pelo utilizador, foi desenvolvido um *script* que permite definir para onde se quer que o veículo autónomo se dirija, o qual deve contornar (na mesma) qualquer limitação de espaço disponível.

Para isso, usando como variável de entrada a pose final, a qual irá ser comparada numa tabela de decisão em relação à sua pose inicial, pode ser determinada, assim, a manobra que deve efetuar. Portanto, até agora, a manobra de rotação consiste somente numa alteração de orientação enquanto que o deslocamento lateral se traduz numa mudança de posição, em particular no eixo dos yy.

Assim, o veículo irá efetuar uma de três operações, apresentadas de forma sucinta na tabela de decisão 5.1:

Algoritmo 1	**Algoritmo 2**	**Algoritmo 3**
$(x,y)_{inicial} \neq (x,y)_{final}$	$\theta_{inicial} \neq \theta_{final}$	$Pose_{inicial} \neq Pose_{final}$
Translação Lateral	Rotação pura	Deslocamento Composto

Tabela 5.1: Fatores de decisão do algoritmo a executar para cada momento da movimentação.

Caso se pretenda uma variação de posição (x,y) deve-se utilizar o algoritmo 1. No caso do objetivo ser a alteração da orientação final, o melhor algoritmo é o 2.

Tendo esta nova resolução de manobras, resta a definição de movimentação no caso de deslocamentos compostos, aqui nomeado algoritmo 3 que foi iniciado mas não completado no decorrer do projeto.

Nas Figuras de 5.8 é ilustrada a composição de duas manobras múltiplas.

Numa primeira iteração, representada na Figura 5.7, o veículo muda a orientação em 180°, necessitando para tal de 2 manobras para o efetuar.

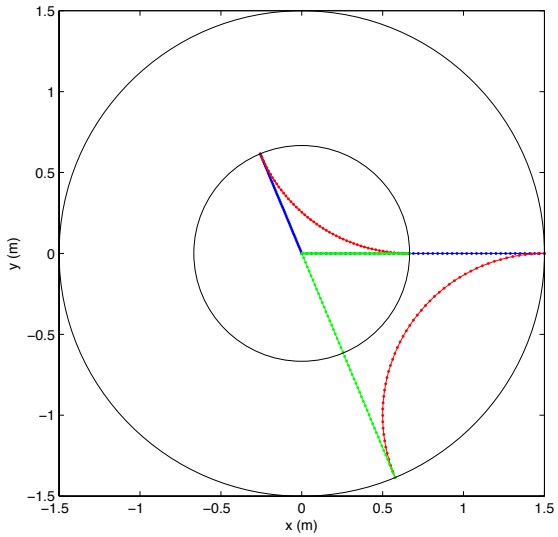
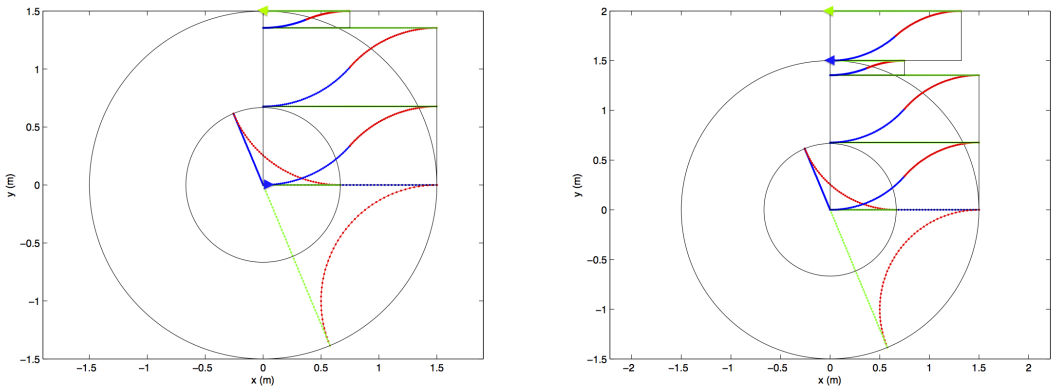


Figura 5.7: Realização de uma manobra de rotação com 180°e raio mínimo de curvatura unitário.



(a) Rotação e translação múltipla para 1.5 *m*. (b) Nova iteração de translação múltipla para 0.5 *m*.

Figura 5.8: Concatenação de manobras múltiplas.

Na Figura 5.8a, o veículo parte com orientação para proceder a uma alteração de posição, neste caso, a 1.5*m* do ponto de partida, executando um total de 3 manobras. No caso de querer prosseguir, como na Figura 5.8b, volta a fazer a decisão e acumulação de manobras múltiplas.

Em ambos os casos é indicada a pose inicial com uma seta azul e a pose final é manifestada pela seta azul.

5.5 Conclusões

As deslocações do veículo autónomo têm que ter sempre em conta o caminho a percorrer, bem como se no mesmo há ou não obstáculos.

Este percurso pode ser feito com recurso a manobras de natureza diferente consoante o objetivo. Quando se pretende que a posição final seja paralela à inicial, utiliza-se uma manobra de translação lateral. No entanto, se se pretende que o veículo apenas mude a sua orientação, procede-se a uma manobra de rotação pura. Sempre que necessário, estas manobras podem ser feitas em conjunto. Para cada uma das situações será necessário aplicar o algoritmo adequado.

Para garantir a integridade do veículo é importante definir previamente o espaço de manobra de forma a que este esteja desimpedido de qualquer obstáculo que possa dificultar a sua realização.

No caso de composição de uma trajetória, resta fazer uma concatenação das manobras múltiplas, dispondo-as de modo a fazer uma movimentação útil em espaços confinados como por exemplo tanques ou outro espaço restrito dentro das mesmas linhas de espaço livre.

Capítulo 6

Conclusões e Trabalho Futuro

O caminho percorrido no âmbito deste trabalho permitiu concluir que os estudos efetuados não são ainda definitivos, pelo que a continuação dos mesmos permitirá obter resultados mais contundentes de acordo com o pretendido para a evolução efetiva da utilização do planeamento de caminhos em diferentes contextos destes veículos autónomos.

6.1 Cumprimento dos Objetivos

Sendo o objetivo principal deste trabalho a caraterização de manobras de uma embarcação autónoma tanto em espaços abertos como restritos, foi feita uma pesquisa sobre os métodos mais comuns e utilizados em veículos semelhantes.

A partir daí, passou-se à aplicação destes métodos às embarcações autónomas nos diferentes ambientes em que elas se movem. Daqui, resultaram vários modelos de movimentos que podem compor diferentes trajetórias.

O trabalho teve início com a pesquisa dos caminhos mais viáveis a que se seguiu a definição de manobras adequadas para um dado percurso.

Sabendo que um veículo não holonómico não desliza lateralmente, nem roda sobre si próprio, foi necessário definir quais as manobras fundamentais para que o mesmo se desloque de um ponto inicial até ao objetivo recorrendo a movimentos elementares. Estes devem ser sempre escolhidos de acordo com o que se pensa ser o caminho mais económico de forma a se dar o mínimo de instruções possíveis ao veículo. Uma manobra pode ser composta por dois arcos de circunferência e um segmento de linha reta. A vantagem do método é que engloba tanto a movimentação paralela ao posicionamento pretendido como a diagonal.

Foi possível executar para o veículo autónomo, que a cada momento, consoante a alteração de pose pretendida e o espaço disponível, decida qual a manobra a efetuar.

A implementação das manobras atrás referidas mostrou-se extremamente útil e bem sucedida, no ambiente de simulação MatLab® .

6.2 Trabalho Futuro

Os resultados apresentados têm por base um cenário ideal.

No entanto, de futuro, o trabalho deverá desenvolver, com mais precisão, as trajetórias de um veículo autónomo. Para tal, torna-se necessário corrigir o erro da pose. Para o efeito, poderá ser utilizado um ponto de referência, a obter através de um sistema de visão, ou então, de um outro sensor, como por exemplo o sistema rádio que o MARES possui entre outros métodos que permitem atualizar a pose do robot com vista à sua verdadeira localização e orientação. Com esta informação, o veículo pode comparar os dados reais com os computados e fazer ele próprio as correções de caminho necessárias para atingir o ponto alvo.

O projeto revela ainda a necessidade de maior planeamento de caminhos. Isto alia-se aos processos iterativos de pesquisa mencionados em 2.4.3 o que permite uma definição mais precisa do melhor caminho a percorrer, mesmo na presença de obstáculos. Por outras palavras, um caminho deixa de ser admissível se interceta com obstáculos, sendo necessário, então, explorar as alternativas viáveis.

Apesar de se terem experimentado outras técnicas de simulação gráficas, verificou-se que a mais adequada para este trabalho seria o MatLab® devido à compatibilidade interplataformas. Para além disso, este ambiente possui uma ferramenta de conversão para linguagem *C* que facilitará a adaptação do código para o veículo autónomo presente no laboratório da faculdade, algo que não foi possível e seria, portanto, o passo seguinte do trabalho.

Anexo A

Instalação do *ROS Fuerte* no Sistema Operativo *OSX*

A linguagem utilizada na instalação deste programa, assim como a resolução de problemas, é internacionalmente acordada que seja o inglês. Como tal, apresenta-se, de seguida, soluções a problemas que poderão ser comuns na instalação do *ROS Fuerte* em *Mac OSX Lion*, sendo que se preveem resultados semelhantes em outras versões do *ROS* e do mesmo sistema operativo.

Estas soluções foram apresentadas também na página *web* da dissertação, tendo sido posteriormente consultadas por endereços de universidades em todo o mundo.

A.1 About *ROS*

A maior frustração em instalar o *ROS Fuerte* [24], encontra-se no facto do guia de instalação não estar completo para os sistemas *OSX*. Este processo, é vagamente baseado na suposição de que o sistema *Unix* tem uma *framework* semelhante à do *OSX*. No entanto, como as bibliotecas são algo diferentes, não se consegue compilar as dependências do sistema.

Installation Procedure

```
rosdep install UWSim ERROR: the following packages/stacks could not have
    their rosdep keys resolved to system dependencies: UWSim: No
    definition of [libopenscenegraph] for OS [osx]
```

Start by opening the *Terminal* and side by side, the *ROS* installation tutorial [24]. The following problems might occur and thus it will be sought a solution to them as they appear on the screen.

1. From the beginning, the tutorial asks the user to write lines in the *.bashrc* file, except that the terminal, each time you close and re-open a window, will not call said file, and yes I did

go to *Terminal* > *preferences* > *shell* which is opened by the command */bin/bash*. So, in order to solve this, the user should write instead in *.profilesource* */.bashrc*.

Furthermore, it is described in [25] that *OSX* doesn't read *.bashrc* file on bash start. Instead, it reads the files, in the following order:

```
/etc/profile
~/.bash_profile
~/.bash_login
~/.profile
```

2. If, for any reason, one of the instructions fail to install because of denied permissions, as it will probably do when the instruction *easy_installpip* is called, the user can *force* it by adding the instruction *sudo* before.

3. When you are installing dependencies, *log4cxx* and *swig-wx* might become a nuisance.

   (a) This might not be a very satisfactory solution but to dig up on *github community*, one will probably not find a better solution. The snag is in:

   ```
   system './autogen.sh'
   ```

   and it fails to install *log*4*cxx*. What the user should do is, comment this line on the formula located at:

   ```
   (/usr/local/Library/Formula/log4cxx.rb)
   ```

   and the it will install just fine.

   (b) Actually *swig* − *wx* has been updated by *Homebrew* so if it fails to install just tun the command again and it should go smoothly. If not, one ought go to this website [26], in which the author displays a a crafty solution developed to this error.

4. Then, it will probably fail to run the instruction *make* − *j*8. This has to do with the *boost* dependency. There isn't a better explanation but, if it snags, the solution is to downgrade this dependency (boost 1.50.0 to boost 1.49.0):

   ```
   brew uninstall boost pushd $(brew --prefix) git checkout -f bf4f9aa
       Library/Formula/boost.rb popd brew install -v boost
   ```

5. It is quite confusing about the paths used to install the libraries. While it reads on the tutorial installation that the user should send said files to */opt/ros/fuerte*, the commands show up */ros*. This location will send the files to *User*.

When installing *ros-underlay* to $/opt/ros/fuerte$, the user should do the *Desktop Full-Install* on this same path so that when the needed command goes $rosdepinstall -a$ it will not turn up the "missing dependencies" error.

6. To do $rosdepinstall - amatplotlib$ installation, this will also return an error. Since the suggested proceeding will not work $(OSX10.7, Python2.7.2 via Homebrew)$, the user has to modify the command as it follows:

```
pip install -e
    git+https://github.com/matplotlib/matplotlib.git#egg=matplotlib
```

7. Finally, sourcing can also crash at one point if the user tries to do any experiment during the installation. One solution is:

   (a) make sure $/usr/local/bin$ is before $/usr/bin$ in

   ```
   $PATH (Lion: > /etc/paths)
   ```

   (b) install $gnu - sed$ with *Homebrew*

   ```
   brew install gnu-sed
   ```

   (c) link $/usr/local/bin/sed to /usr/local/bin/gsed$

   ```
   (ln -s /usr/local/bin/gsed /usr/local/bin/sed)
   ```

A.2 About *UWSim*

Este simulador acaba por não ser implementado devido à falta de compatibilidade de bibliotecas e também, de placas gráficas. Para além disso, o investigador chefe, com o qual foram trocadas conversações sobre o assunto, admitiu não existir documentação sobre o simulador, deixando o progresso muito à mercê de tentativas e erro. Sendo assim esta abordagem acaba por ser abandonada no âmbito desta dissertação.

The main problem with *UWSim*, is that it doesn't have a rule for the *libopenscenegraph* package for *OSX*, so, since it wasn't appealing to struggle yet with every other little solution one would have to try, the user should just move on and work tutorials of *ROS* in the **Ubuntu** environment, this has been done with *12.04 Precise* and the following has been addressed:

When running *UWSim*, it may reveal strange visualization effects, or it may crash before it even starts. As referred before, it probably means that your board does not meet the requirements (if this is the case, there is a $--disableShaders$ option). A list of cards that are known to be working are displayed in the simulator's website.

By going through the tutorials one by one, it seems as though that while *ROS Fuerte* was build to be boulder and more efficient cruising directories, there's still some bugs to treat. They don't really interfere with the tutorials but one of the most pertinent error is revised:

There is no turtle on the first blue screen. It is possible that the installation is not properly devised. Or the *ROS_PACKAGE_PATH* is not correct. It's of somewhat simple direction, defining environment variables. The user can open this path and define it there directly, as it is being used. If, on the other hand, it's not needed anymore, then one can open it and just delete the unwanted path.

A.2.1 About *Blender*

The XML file describes the general scenario, and simulation parameters. However, robots are described with an *URDF* file. A scene XML file may make a reference to an *URDF* file each time it includes a robot into the scene.

Creating new scenes is done by writing a XML file. To make *UWSim* load a scene, simply run the $--configfile$ option, e.g:

```
rosrun UWSim UWSim --configfile path/to/yourScene.xml
```

Considerations

Sometimes, in the XML file, it will point to an *.osg* file type. This is primarily associated with *OpenSceneGraph*, by the *OSG Community*.

Instead of using *.osg* it would be more accessible (since it is an opensoftware) the 3D modeling program called *Blender*. This produces a *blender* file that is easily exported as an *.obj* file readable by *OSG* or in any of the formats supported: *.osg, .ive, .stl, .3ds, etc*.

Referências

[1] Aníbal Castilho Coimbra de Matos. Modelo de um robot móvel. Notas de aula de Sistemas Robóticos Autónomos, 2011. Faculdade de Engenharia da Universidade do Porto.

[2] Investigador Chefe Cameron Riviere, Projeto iniciado em 2003. URL: `http://www.cs.cmu.edu/~heartlander/rob`.

[3] Investigadora Chefe Cecilia Laschi. Projeto coordenado pela Scuola Superiore Sant ´Anna (Pisa, Itália), 2009. URL: `http://www.octopusproject.eu/about.html`.

[4] Jianhua Wang, Pingping Huang, Changfeng Chen, Wei Gu, e Jianxin Chu. Stereovision aided navigation of an autonomous surface vehicle. Em *Advanced Computer Control (ICACC), 2011 3rd International Conference on*, páginas 130–133, 2011. doi:10.1109/ICACC.2011.6016382.

[5] Mark W Spong, Seth Hutchinson, e Mathukumalli Vidyasagar. *Robot modeling and control*. John Wiley & Sons New York, 2006.

[6] D. Bodhale, Nitin Afzulpurkar, e N.T. Thanh. Path planning for a mobile robot in a dynamic environment. Em *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, páginas 2115–2120, 2009. doi:10.1109/ROBIO.2009.4913329.

[7] Qiang Zhan, Shouren Huang, e Jia Wu. Automatic navigation for a mobile robot with monocular vision. Em *Robotics, Automation and Mechatronics, 2008 IEEE Conference on*, páginas 1005–1010, 2008. doi:10.1109/RAMECH.2008.4681418.

[8] Johann Borenstein, Hobart R Everett, Liqiang Feng, e David Wehe. Mobile robot positioning – sensors and techniques. Relatório técnico, DTIC Document, 1997.

[9] Roland Siegwart, Illah R Nourbakhsh, e Davide Scaramuzza. *Autonomous mobile robots*. Prentice Hall India, segunda edição, 2011.

[10] Mark H Overmars. A random approach to motion planning. *RUU-CS*, (92-32), 1992.

[11] Gregory Dudek e Michael Jenkin. *Computational principles of mobile robotics*. Cambridge university press, 2010.

[12] Jean-Claude Latombe. *Robot motion planning*. Kluwer Academic Publishers (Boston), 1991.

[13] Steven Michael LaValle. *Planning algorithms*. Cambridge university press, 2006.

[14] José Miguel Soares de Almeida. Controlo híbrido de robots não holonómicos. Tese de mestrado, Faculdade de Engenharia da Universidade do Porto, 1996.

[15] J. Barraquand e J.-C. Latombe. On nonholonomic mobile robots and optimal maneuvering. Em *Intelligent Control, 1989. Proceedings., IEEE International Symposium on*, páginas 340–347, 1989. doi:10.1109/ISIC.1989.238696.

[16] Ju. I. Neimark e Nikolaĭ A. Fufaev. *Dynamics of nonholonomic systems*, volume 33. American Mathematical Soc., 1972.

[17] Francesco Bullo, N. Ehrich Leonard, e Andrew D. Lewis. Controllability and motion algorithms for underactuated lagrangian systems on lie groups. *Automatic Control, IEEE Transactions on*, 45(8):1437–1454, 2000.

[18] URL: `http://wiki.ros.org/`.

[19] J.; Fernandez J.J.; Sanz P.J. Prats, M.; Perez. An open source tool for simulation and supervision of underwater intervention missions. Em *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, páginas 2577–2582, 7-12 October 2012.

[20] URL: `http://oceansys.fe.up.pt/tech_mares.php`.

[21] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[22] J. A. Reeds e L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990. URL: `http://projecteuclid.org/DPubS?service=UI&version=1.0&verb=Display&handle=euclid.pjm/1102645450`.

[23] F Gómez-Bravo, F Cuesta, e A Ollero. Parallel and diagonal parking in nonholonomic autonomous vehicles. *Engineering Applications of Artificial Intelligence*, 14(4):419–434, 2001.

[24] URL: `http://wiki.ros.org/fuerte/Installation/OSX/Homebrew/Source`.

[25] URL: `http://apple.stackexchange.com/questions/12993/why-doesnt-bashrc-run-automatically#comment13715_13019`.

[26] URL: `http://ardadv.blogspot.pt/2012/05/using-ros-robot-operating-system-fuerte.html?showComment=1341275868632#c5264086083107923782`.